DETECCIÓN DE ANOMALÍAS EN DATOS DE INSPECCIÓN DE VEHÍCULOS

David Salcedo Gutiérrez

MÁSTER EN INGENIERÍA INFORMÁTICA
FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID

Trabajo Fin Máster

Septiembre 2018
Curso Académico: 2017/2018

Director: Rafael Caballero Roldán

Calificación: 6

Agradecimientos

Quiero agradecer a mi familia el apoyo y los ánimos dados, ya que sin ellos no habría sido posible llegar hasta aquí.

Agradecer mi director de Trabajo Final de Máster Rafael Caballero Roldán por permitirme trabajar en este proyecto, por la ayuda y guía suministrada en el mismo.

Resumen en castellano

El objetivo de este TFM ha sido desarrollar una aplicación que sea capaz de obtener la información necesaria para discernir si una inspección de ITV tiene anomalías en sus resultados o no a partir de una fuente de datos introducida por texto o Excel.

En particular, nos interesa detectar si diferentes inspectores dentro de la misma ITV actúan de forma diferente. Para ello partiremos de un fichero de datos con los datos de las inspecciones durante un periodo relativamente largo (por ejemplo, un año).

Con este propósito nos hemos servido del cálculo de reglas de asociación mediante el algoritmo *Apriori*. Las reglas de asociación nos permiten saber qué deficiencias de los vehículos se encuentran habitualmente de forma conjunta durante la inspección de un vehículo.

Usando los resultados de las reglas de asociación junto con los test estadísticos, podremos verificar si hay alguna discrepancia entre los resultados obtenidos por cada inspector en las ITV.

El resultado del trabajo es un método que permite detectar estas anomalías, mostrar los defectos que atestiguan el comportamiento anómalo y, lo que es más importante, asegurar que las anomalías detectadas son estadísticamente significativas. Este método ha sido implementado en una herramienta disponible que puede ser empleada por entidades de inspección de ITVs como ENAC.

Palabras clave

Defectos en vehículos, test estadístico, ITV, inspector, Apriori, reglas de asociación, anomalías, transacciones.

Resumen en inglés

The objective of this TFM has been to develop an application that can discern if an ITV inspection has anomalies in its results from a data source entered by text or Excel.

In particular, we are interested in detecting if different inspectors within the same inspection station act differently. The dataset considered consists of data files containing the inspection data for a relatively long period (e.g. one year).

To do this, we have used the association rules using the *Apriori* algorithm. The association rules allow us to know which deficiencies of the vehicles are usually found together during the inspection of a vehicle.

Using the results of the association rules together with the statistical tests, we will be able to verify if there is any discrepancy between the results obtained by each inspector in the ITV.

The result of the work is a method to detect these anomalies, to show the defects that testify to the abnormal behavior and, more importantly, to ensure that the anomalies detected are statistically significant. This method has been implemented in an available tool that can be used by ITVs inspection entities such as ENAC.

Keywords

Defects, Statistical Tests, ITV, inspector, Apriori, Association rules, anomalies, vehicle, transaction.

INDICE

Índice de figuras

Índice de Tablas

1. Introducción

1.1. Contexto inicial

La minería de datos [4, 32] es un campo de la estadística y las ciencias de la computación. Incluye un conjunto de técnicas y tecnologías que permiten explorar grandes bases de datos, de manera automática o semiautomática, con el objetivo de encontrar patrones repetitivos, tendencias o reglas que expliquen el comportamiento de los datos en un determinado contexto.

Algunas de las técnicas de la minería de datos son los grupos de registros de datos (análisis clúster), registros poco usuales (la detección de anomalías) y dependencias (minería por reglas de asociación). Esta disciplina combina técnicas típica de campos como la de la inteligencia artificial, el aprendizaje automático, la estadística y los sistemas de bases de datos [25].

En este trabajo utilizamos en particular la técnica propia de minería de datos conocida como *reglas de asociación*, en particular en su implementación mediante el algoritmo *Apriori*.

Otro pilar sobre el que se sustenta este proyecto es la unión entre el uso de reglas de asociación y test estadísticos, estos últimos permiten analizar los datos obtenidos de las reglas de asociación y comparar entre inspectores para poder analizar si hay alguna anomalía en las ITV (Inspección Técnica de Vehículos).

Link GitHub: https://github.com/Dsalce/TFM.git

1.1.1. Objetivos y motivación

Antes de entrar en este apartado es necesario comentar que las ITVs se encargan de inspeccionar vehículos y que a su vez las propias ITVs son auditadas por entidades como ENAC (Entidad Nacional de Acreditación) que tratan de determinar si se adecuan a los estándares de calidad requeridos.

Los auditores tienen la necesidad de señalar las posibles anomalías de forma detallada y de eliminar en la medida de lo posible el factor azar, es decir que las anomalías deben referirse a diferencias entre los datos estadísticamente significativas.

El proyecto descrito no está basado en ningún otro proyecto desarrollado en la Universidad Complutense de Madrid (UCM) o en algún otro proyecto hasta donde hemos podido comprobar. La combinación de reglas de asociación con los test estadísticos hace bastante único a este proyecto.

La motivación principal de este proyecto surge por la necesidad de agilizar el proceso de analizar si en las ITV realizadas por los inspectores se produce alguna anomalía. Para detectar esto se va a comparar los resultados que tiene cada inspector en las ITV con el resto, pudiendo así verificar que las diferencia entre los datos son estadísticamente significativas.

Otro de los objetivos es verificar si hay relación entre los defectos detectados, es decir queremos verificar si hay defectos que siempre o casi siempre aparecen emparejados con otro defecto, para así poder asegurar que si aparece un defecto determinado es casi seguro que va a aparecer el otro defecto emparejado.

El principal objetivo a nivel personal es adquirir conocimientos de estadística y aprender a aplicar esos conocimientos en un entorno de desarrollo, así como mejorar en el lenguaje de programación de Python y aprender acerca del campo de la minería de datos que cada vez es más relevante en el mundo tecnológico y seguirá aumentando.

1.1.2. Plan de trabajo

El proyecto se ha divido en varias etapas. La primera etapa constó del análisis de los ficheros proporcionados por ENAC y parseo de los mismos para adaptar los datos a un formato legible, este fue un trabajo arduo por el formato especial de los ficheros.

Los ficheros contaban con un formato de texto bastante particular, lo que obligaba a realizar un análisis especifico de los datos, por ejemplo, para obtener las longitudes de las cabeceras era necesario leer la línea siguiente del fichero, la cual estaba compuesto por guiones que correspondían con la longitud de cada cabecera, una vez procesada esta línea ya se podía realizar la obtención de las cabeceras.

Estos ficheros también contaban con caracteres especiales y en función de su procedencia el formato no era exactamente igual lo que dificultaba en gran medida implementar un algoritmo común para el procesamiento de todos los ficheros.

Por último, se realizó el proceso de anonimización, el cual consistió en quitar todos aquellos datos que fuesen personales para cumplir con el Reglamento General de Protección de Datos (RGPD), como el número de bastidor del coche, matricula, nombres y apellidos, etc.

Una vez completada la primera etapa, se empezó con el desarrollo de la aplicación y el estudio de las reglas de asociación para esclarecer si las podíamos usar para este proyecto. Cuando se demostró que si iban a dar resultado se realizó su inclusión en la aplicación y explicación en la memoria.

En la tercera etapa se realizó el estudio de los test estadísticos que iban a poderse usar en el proyecto, una vez terminado el estudio se incluyó su uso a la aplicación.

Por último, en la cuarta etapa se realizó la escritura de la memoria y la inclusión del epígrafe de los test estadísticos en la misma.

1.1.3. Organización de la memoria

La memoria está estructurada en 7 capítulos:

Primer capítulo: Consta de la Introducción en la que se realiza un resumen de la aplicación.

Segundo capítulo: En este capítulo se habla de las reglas de asociación.

Tercer capítulo: Especificación de las anomalías relacionadas con los inspectores.

Cuarto capítulo: Explicación y uso de los test estadísticos en el proyecto.

Quinto capítulo: Explicación de las tecnologías usadas.

Sexto capítulo: Explicación de las funcionalidades de la aplicación.

Séptimo capítulo: Conclusiones y plan de trabajo futuro.

1.2. Initial context

Data mining [4, 32] is a field of statistics and computer science. It includes a set of techniques and technologies that allow us to explore large databases, automatically or semi-automatically, in order to find repetitive patterns, trends or rules that explain the behavior of data in a given context.

Some of the techniques of data mining are groups of data records (cluster analysis), unusual records (anomaly detection) and dependencies (mining by association rules). This discipline combines techniques typical of fields such as artificial intelligence, automatic learning, statistics and database systems [25].

In this work we use in particular the data mining technique known as *association rules*, In particularly its implementation through the *Apriori* algorithm.

Another pillar on which this project is based on the union between the use of association rules and statistical tests, the last one allowing to obtain the data from the association rules to be analyzed and compared between inspectors in order to analyze whether there are any anomalies in the ITV.

Link GitHub: https://github.com/Dsalce/TFM.git

1.2.1. Goals and motivation

Before going into this section, it is necessary to comment that the ITVs are in charge of vehicle inspection and that the ITVs themselves are audited by entities such as Enac that try to determine if they adapt to the required quality standards.

Auditors need to point out possible anomalies in detail and eliminate the random factor as far as possible, i.e. anomalies should refer to differences between statistically significant data.

The project described is not based on any other project carried out at the Complutense University of Madrid (UCM) or on any other project. The combination of association rules with statistical tests makes it quite unique.

The main reason for this project is the desire to speed up the process of analyzing whether any anomaly has occurred in the ITV inspections carried out by the inspectors. In order to detect this, the results of each inspector will be compared with the rest of the inspector's results in the ITVs, thus being able to verify that the differences between the data are statistically significant.

Another of the objectives is to verify if there is a relationship between the defects detected, we want to verify if there are defects that always or almost always appear paired with another defect, so we can assure that if a certain defect appears it is almost certain that the other defect will appear paired.

The main objective at the personal level is to acquire statistical knowledge and learn how to apply that knowledge in a development environment, as well as to improve my skills in Python and learn about the field of data mining that is becoming increasingly relevant in the technological world and will continue to grow.

1.2.2. Work Plan

The project has been divided into several stages. The first stage consisted of analyzing the files provided by Enac and parsing them to adapt the data to a readable format, this was an arduous task due to the special format of the files.

The files had a very particular text format which required a specific analysis of the data, for example, to obtain the lengths of the headers it was necessary to read the following line of the file, which was composed of dashes that corresponded to the length of each header, once this line had been processed it was already possible to obtain the headers.

These files also had special characters and depending on their origin, the format was not exactly the same, which made them very difficult to implement a common algorithm process to all the files.

Finally, the anonymization process was carried out, which consisted of removing all personal data in order to comply with the General Data Protection Regulation RGPD, such as the car's chassis number, registration number, first and last names, etc.

Once the first stage was completed, we began with the development of the application and the study of the rules of association to clarify if we could use them for this project. When it was demonstrated that if they were going to work, they were included in the application and explained in the report.

In the third stage, the statistical tests that could be used in the project were studied, and once the study was finished, their use was included in the application.

Finally, in the fourth stage, the report was written, and the statistical tests were included in the report.

1.2.3. Report organization

The report is structured in 7 chapters:

Chapter first: It consists of the Introduction in which a summary of the application is made.

Chapter Two: This chapter discusses the rules of association.

Chapter third: Specification of anomalies by inspector.

Chapter Fourth: Explanation and use of the tests in the project.

Chapter Fifth: Explanation of the technologies used.

Chapter Sixth: Explanation of the application's functionalities.

Chapter Seven: Conclusions and future work plan.

2. Reglas de asociación

Las reglas de asociación [4, 23] constituyen una técnica de minería de datos orientada a la obtención de relaciones entre los elementos de conjuntos de datos generalmente muy amplios. En definitiva, la idea general consiste en encontrar aquellos datos que muy a menudo van asociados a otros.

Uno de los usos que se le ha dado tradicionalmente a las reglas de asociación es el del "análisis de la cesta de la compra" (Market Basket Analysis [7, 15]). Vamos a repasar este caso de uso ya que resulta muy útil para entender el concepto de regla de asociación.

Esta aplicación consiste en analizar qué productos aparecen en la cesta de la compra junto con otros. Usando esta información, las empresas, en este caso los supermercados, pueden decidir dónde colocar sus productos para poder maximizar sus beneficios, ya sea posicionando productos similares cerca o bien haciendo a las personas recorrer más tramo del supermercado para que visualicen más productos.

Un ejemplo de compras realizadas podría ser el que se muestra en la tabla 2.1.
Cada fila corresponde a los productos de un carro de la compra de un usuario del supermercado.

TID	Instancia
1	{Queso, Leche}
2	{Pan, Queso, Leche}
3	{Pan,Queso,Leche,Patatas,Pizza}
4	{Leche,Patatas,Vino,Jamón}

Tabla 2.1: Tabla de transacciones.

Cada uno de estos conjuntos se denomina una transacción. Las transacciones son por tanto subconjuntos del conjunto total de posibles elementos. Por ejemplo, en el conjunto de transacciones de la figura, observamos que siempre que está incluido el elemento *Queso*, también aparece el elemento *Leche*. Esta relación es a la que denominaremos regla de asociación, y la escribiremos con la forma {Queso}→{Leche}. A continuación, definimos más formalmente estos conceptos.

2.1. Medidas de las reglas de asociación y notación

Partimos de un conjunto de datos inicial I (I En el ejemplo mostrado en el apartado anterior correspondería con todos los productos del supermercado). Una transacción está formada por un subconjunto de elementos de I, llamado *instancia*, y además elementos adicionales como el identificador o el orden de la transacción dentro del total de transacciones. Suponemos un conjunto finito de transacciones *T = {t1, ...tn}*. Algunos autores llaman a T *conjunto de entrenamiento* y otros *bases de datos*.

Para definir una regla de asociación tomamos subconjuntos no vacíos de I, llamados comúnmente X, Y:

X corresponde con el lado izquierdo de la regla.

Y corresponde con el lado derecho de la regla.

La forma general de expresar una regla de asociación es: X→Y. Se denomina *itemset* de una regla de asociación al conjunto de elementos o ítems de I que aparecen en dicha regla, es decir a X U Y.

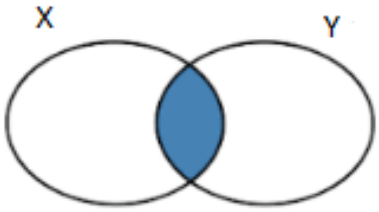


Figura 2.1: Reglas de asociación [4].

En una regla podemos señalar las siguientes medidas básicas.

- **NX**: Es el número de instancias que contienen X, esto es el número de transacciones que incluyen los elementos en X.

- **NY**: Número de instancias que contienen Y.

- **NBOTH**: Número de instancias que contienen a X e Y.

- **NTOTAL**: Número total de instancias, esto es, número de transacciones considerado.

En esta nomenclatura seguimos los nombres habituales utilizados en los textos de minería de datos [4].

Los valores descriptivos anteriores se usan principalmente para calcular otras medidas más interesantes:

- **Soporte(X)**
  Dado un conjunto de elementos X, el soporte [42] es la proporción de transacciones en T que incluyen X. Se representa como sop(X).

- **Confianza(X→Y)**
  La confianza [43] de una regla es la proporción dado el conjunto de elementos de T que contienen X, que contienen también una proporción Y. La confianza se representa como conf(X→Y).
  $$conf(X \rightarrow Y) = sop(X \cup Y) / sop(X).$$

- **Lift (X→Y) = conf(X→Y) / sop(Y) = sop(XUY)/(sop(X) x sop(Y))**
  El lift [11] es un valor que nos puede ayudar a diferenciar si una regla encontrada es interesante o no, indica si los componentes de una regla aparecen frecuentemente juntos o no.

  - Si lift(X→Y) > 1 indica que ambos componentes de la regla aparecen más frecuentemente juntos de lo que se esperaba.

  - Si lift(X→Y) < 1 indica que ambos componentes de la regla aparecen menos frecuentemente juntos de lo que se esperaba.

- Si lift(X→Y) = 1 indica que ambos componentes de la regla aparecen juntos las mismas veces que se esperaba.

2.2. Generar Reglas de asociación

Dado que el número de posibles reglas de asociación es muy elevado, se emplean diversos criterios para limitar la generación a aquellas reglas que pueden resultar de interés. Una posibilidad es generar todas las reglas posibles a partir de un conjunto de datos que tengan un soporte y confianza superiores a unos valores mínimos a los que se denomina *minsop* y *minconf*.

La forma más simple, aunque ineficiente comparada con otro algoritmo que veremos posteriormente consta de dos etapas.

1. La primera etapa consiste en generar todos los itemset de X U Y en los que se cumpla que *sop(x)>minsop*.

2. La segunda etapa consiste generar todas las posibles reglas para un itemset con al menos un elemento en cada lado de la regla que cumplan con que la *conf(X→Y)> minconf.*

Es importante destacar es que la cantidad de reglas que se puede crear para un conjunto de datos pequeño es enorme 2^k-1, siendo k = |T|, pero hay algunas que son más significativas que otras ya que tienen más repercusión en el conjunto de datos.

Una vez calculadas las reglas y sus medidas se pueden usar las medidas para discernir si una regla es interesante para nuestros propósitos o no, estableciendo unos criterios determinados siendo frecuente usar en el caso del soporte *minsop* y en el caso de la confianza *minconf* como medida para filtrar las reglas.

Por ejemplo, si tenemos un conjunto de datos de una lista de la compra con el requisito de *minsop*>0,3 (como el que hemos visto en la sección del comienzo de este capítulo). Obtenemos una regla que relaciona el Pan→Leche con el sop(X)=0,2 esta regla no nos interesara y por tanto no la tendremos en cuenta como regla, esto debido a que no cumple con el requisito por ser *minsop<sop(X).*

Otra forma de generar reglas de asociación de forma más eficiente es el método de Apriori, que describimos a continuación.

2.3. Apriori

El algoritmo Apriori fue creado por Agrawal and Srikant [44] con el cual demostró que se podían crear reglas de asociación en un espacio de tiempo bastante más reducido que generando todas las posibles reglas. El algoritmo en primer lugar calcula las partes izquierdas de las reglas y en una segunda fase las partes derechas.

El algoritmo se basa principalmente en los siguientes teoremas:

- **Si un itemset es frecuente los subitems o subconjuntos no vacíos también deben ser frecuentes.**

  La frecuencia de un subconjunto debe ser mayor o igual que el del propio conjunto, por lo tanto, se deduce que cualquier subconjunto (no vacío) de un elemento admitido también debe ser compatible.

- **Si un conjunto de datos con cardinal k es vacío los conjuntos de datos generados k+1, k+2, etc serán también vacíos.**

  Un conjunto k+1 o mayor incluye al subconjunto k, si el conjunto de k es vacío, k+1 en consecuencia también deberá ser vacío y por lo tanto lo mismo sucederá con los de tamaño k+2 y sucesivos.

- **Transferir elementos del itemset de la parte izquierda de la regla a la parte derecha no puede incrementar la confianza de la regla.**

  Si tenemos una regla X U Y→Z transferir Y al lado derecho obteniendo X→ Y U Z no puede significar un aumento en la confianza de la regla.

Los teoremas del algoritmo Apriori se pueden obtener de Bramer, M. A. (2017). *Principles of data mining* [4].

Para comprender mejor el algoritmo Apriori, vamos a realizar un ejemplo con un conjunto de elementos {a,b,c,d,e} para este itemset nos saldrán todas las combinaciones de itemset que se ven en la figura 2.2. De todas las reglas que obtenemos nos quedaremos con aquellas que tengan mayor frecuencia y podamos todos aquellos que sean infrecuentes.

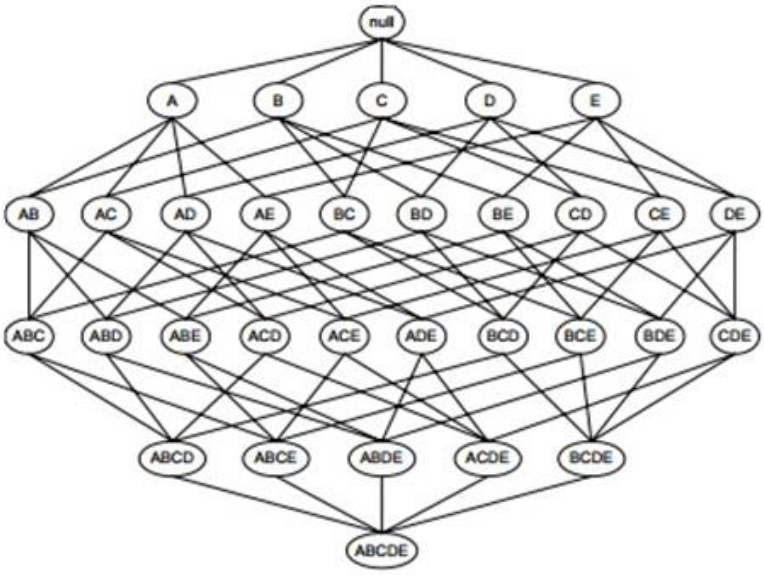


Figura 2.2: Árbol de reglas [7, 19].

Suponemos que AB es itemset poco frecuente, por consiguiente, todos los itemset donde se encuentre este conjunto de datos también serán poco frecuentes. Por lo tanto, podamos todos los itemset que incluyen AB ignorándolos para obtener las reglas de asociación como se muestra en la figura 2.3.

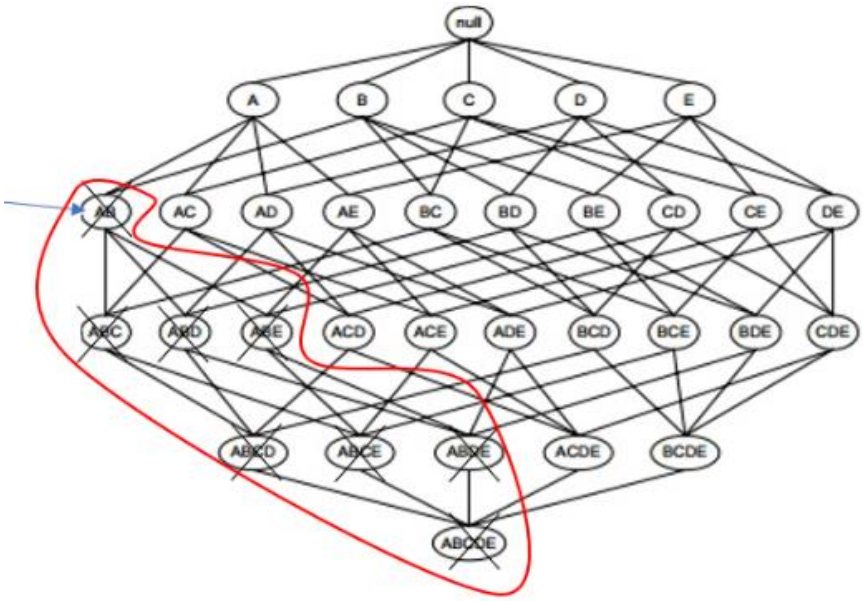


Figura 2.3: Árbol podado [7, 19].

2.3.1. Secuencia algoritmo Apriori

El algoritmo Apriori se puede dividir en dos pasos que se repiten sucesivamente como se muestra a continuación.

Antes de empezar con los pasos es importante definir la regla *self-join* [31] la cual usaremos más adelante. Esta regla consiste en la unión de parejas que contengan el mismo elemento en los itemset, por ejemplo, supongamos que hemos obtenidos dos itemset {ab} y {ad}.

Obtenemos el elemento que se repite en ambos itemset en este caso "a", y lo juntamos con los otros dos elementos que acompañan a "a" en sus respectivos itemset, el resultado que se obtiene es {abd}.

1. Obtención del sop(x) de todos los itemset del conjunto de datos y comprobación de que cumplen *sop(X)>minsop*, si no cumplen son elementos infrecuentes serán podados y no se tendrán en cuenta.

2. Uso de la regla *self-join*, para encontrar las itemset que cumplan *sop(X)>minsop* con k+1 elementos. Este proceso se repetirá hasta que no se pueda aplicar la regla *self-join*.

En el diagrama de flujo que se muestra a continuación se ve de una forma más detallada cuál es la secuencia que se sigue para calcular las reglas de asociación mediante el algoritmo Apriori.

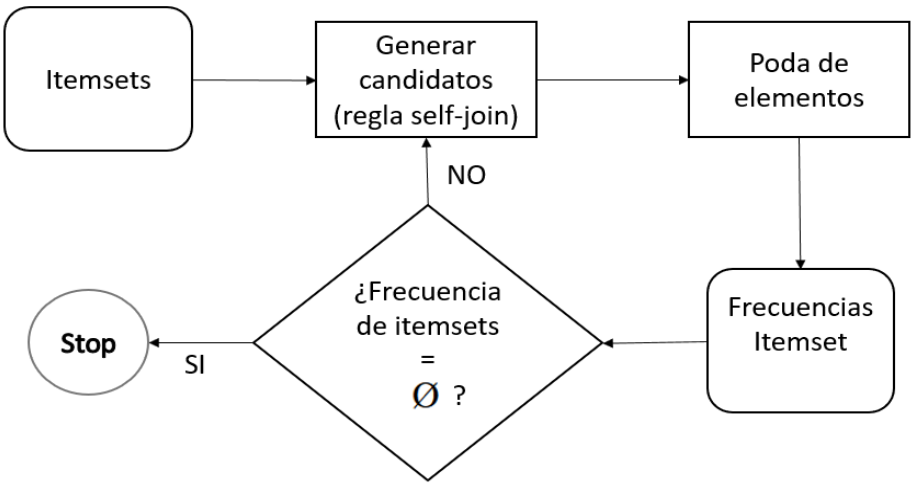


Figura 2.4: Diagrama de actividad [3, 8, 31].

No obstante, a continuación, vamos al realizar un ejemplo para que quede más clara la explicación.

En el ejemplo partimos de un conjunto de datos que consta de 5 transacciones las cuales contienen un conjunto de elementos tal como se muestra en la tabla 1 (figura 2.5).

Hemos establecido que las frecuencias que deben cumplir los elementos son de minsop=0,4, con lo cual el *sop(x)>minsop*.

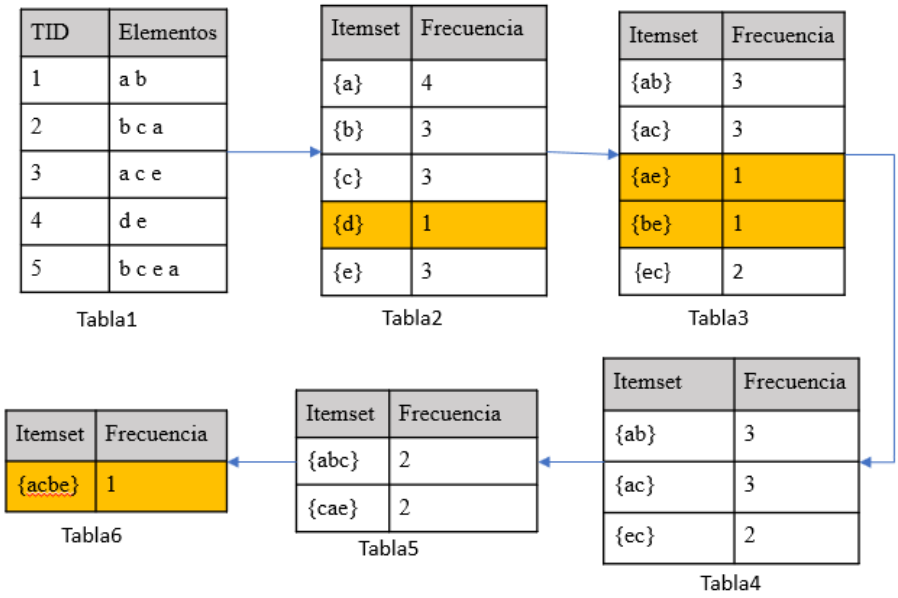


Figura 2.5: Ejemplo Apriori [8, 31].

1. El primer paso es obtener las frecuencias del conjuntos unitarios formado por los elementos tal y como se muestra en la tabla 2 (figura 2.5) como el elemento {d} no cumple con el soporte mínimo.

   Para obtener la proporción de un elemento en el conjunto de datos dividimos la frecuencia del mismo entre el total de transacciones. En este caso seria 1/5<0,4, el elemento no cumple con el soporte mínimo por lo tanto lo podamos.

2. En el segundo paso se consideran las parejas formadas por elementos obtenidos en el primer paso, con los elementos resultantes del primer paso calculamos las nuevas frecuencias de las parejas. El resultado que obtenemos es el que se representa en la tabla 3 (figura 2.5).

3. En el tercer paso descartamos los elementos que no cumplan con el *minsop*, que en este caso son las parejas {ae} y{be} y obtenemos el conjunto de datos que se representa en la tabla 4 (figura 2.5), ya que la frecuencia que tienen ambos elementos es inferior al *minsop*, 1/5<0,4.

4. En el cuarto paso obtenemos nuevos conjuntos mediante la regla de *self-join*, de la unión de las parejas obtenemos tríos como se muestra en la tabla 5 (figura 2.5). En este caso {abc} y {cae} tienen una frecuencia de 2/5, la cual es mayor que el *minsop* (0,4), por lo tanto, no hay que descartar ningún trío.

5. Por último, mediante el uso de la regla *self-join* obtenemos la última unión de los tríos {acbe} tabla 6 (figura 2.5). Como {acbe} no cumple con el *minsop<0,4* hemos terminado. Es posible que la última unión sí cumpla con el minsop en este caso seguiríamos hasta que quede el conjunto vacío.

2.3.2. Generación de reglas de asociación

El método explicado en el apartado anterior sirve para generar la parte izquierda de la regla de forma eficiente. Ahora veremos cómo generar la regla de asociación.

Para generar la regla de asociación se realiza un proceso similar al explicado para calcular los elementos más frecuentes del conjunto de datos. Se deben generar todas las partes derechas mediante los elementos de X U Y, la parte izquierda se generará con todos aquellos elementos que no se usen en la parte derecha.

En el caso de las reglas se debe usar la confianza *conf(X→Y)* para discernir si esa regla generada es una regla es frecuente o infrecuente, también se usará una variable con la que estableceremos el mínimo de confianza que deben tener las reglas para ser admitidas *minconf*.

Las reglas que no cumplen con un mínimo de confianza son podadas. Si por el contrario *conf(X→Y)>minconf* no se poda la regla, un ejemplo de poda de la regla es el que se muestra en la figura 2.6.

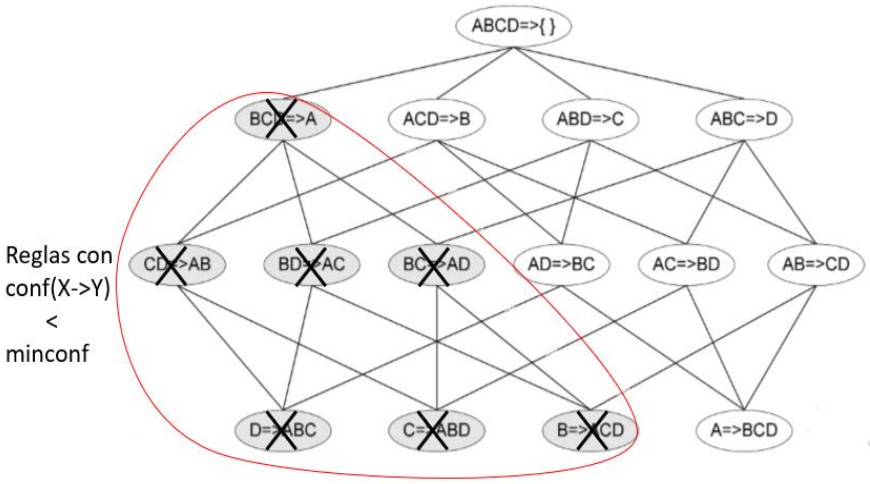


Figura 2.6: Generar reglas de asociación [5, 7].

2.3.3. Pseudocódigo algoritmo Apriori

El algoritmo Apriori consta de los pasos que se muestra a continuación en pseudocódigo de la figura 2.7.

1. Al comienzo del algoritmo (paso 01) se obtienen los itemsets o conjunto de datos para calcular la frecuencia de ocurrencia de los mismos. Al finalizar estos pasos obtendremos los conjuntos de datos frecuentes de la primera iteración.

2. A continuación, el algoritmo cuenta con un "while" (paso 03), mientras haya elementos se continuará realizando iteraciones. En el (paso 04) se generan los candidatos mediante la función Apriori-gen la cual se explicará más en detalle en el siguiente apartado. Después de generar el conjunto de candidatos, se comprueba que el conjunto no es vacío, ya que si no existen candidatos se termina la ejecución del algoritmo (paso 05).

3. Mas tarde se recorren todas las transacciones del conjunto de datos (pasos 06-11). Se obtienen los candidatos de las transacciones (paso 07) y posteriormente se realiza una contabilidad de ocurrencia de los itemset (pasos 08-10).

4. Una vez calculadas las frecuencias de ocurrencia de los elementos el algoritmo poda todos los elementos que no hayan cumplido con el mínimo de soporte(>= minsop) (paso 12).

5. Por último, el algoritmo termina cuando no queden más itemset.

```
01: L_1={l_1,...,l_n|∀l∈large itemsets}  //Obtención de los itemsets
02: k=2
03: while(L_{k-1}!= Ø)
04:   C_k=apriori-gen(L_{k-1})  //Generación de Candidatos
05:   if(C_k=Ø) return endif
06:   for transacciones t ∈ T  //Foreach de todas las transacciones
07:     C_t =subset(C_k,t) //Obtención de candidatos que pertenecen a T
08:     for candidatos c ∈ C_t //Foreach de los candidatos
09:         c.count+=1 //Contabiliza frecuencia
10:     endfor
11:   endfor
12:   L_k={c∈ C_k |  c.count >= minsop} //Extracción de los elementos frecuentes
13:   k+=1
14: end while
```

Figura 2.7: Pseudocódigo algoritmo Apriori [26, 14, 2].

2.3.3.1. Función Apriori-gen

La función Apriori-gen mencionada previamente genera itemsets candidatos a partir del conjunto de datos calculado en la iteración anterior del algoritmo Apriori. Mediante la mezcla de los itemset frecuentes o la denominada regla *self-join* se obtiene un itemset mayor, a continuación, pasaremos a explicar la función a priori-gen la cual esta implementada en la figura 2.8.

1. Al comienzo de la función inicializa C_k como conjunto vacío (paso 02), posteriormente realizará un bucle el cual se mantiene mientras los elementos de los itemset comparados sean iguales (paso 03).

2. A continuación, construimos un itemset mayor a los del conjunto mezclando los valores del conjunto L_{k-1} (paso 04). Más tarde en el (paso 06-07) si los itemset de L_{k-1} están contenidos en el conjunto obtenido en el paso 04 se añaden al conjunto C_k.

3. Por último, se devuelve C_k y se termina la función.

01: Funcion apriori-gen(L_{k-1})
02: C_k= Ø
03: foreach ($L_{k-1}.item_p$,$L_{k-1}.item_q$|$L_{k-1}.item_p[i]$=$L_{k-1}.item_q[i]$,∀i ∈{1,...,k−2})
//Recorre todos los elementos de L_{k-1} mientras sean iguales
04: c=$L_{k-1}.item_p[1]$,...,$L_{k-1}.item_p[k-2]$,$L_{k-1}.item_p[k-1]$,$L_{k-1}.item_q[k-1]$
//Construye un itemset de mayor tamaño
05: if($L_{k-1}.item$⊂ c)
//Si los itemsets están contenidos en c los añade a los candidatos resultantes
06: C_k= C_k∪c
07: endif
08: endfor
09: return C_k
10: endFuncion

Figura 2.8: Pseudocódigo función Apriori-gen [26, 2].

2.4. Resultados: Reglas de asociación

En este apartado analizaremos los resultados obtenidos durante el proceso de obtención de las reglas de asociación en el proyecto.

2.4.1. Descripción del conjunto de datos

Para poder entender los resultados de las reglas de asociación primero debemos de hablar de conjunto de datos. Contamos con 6 ficheros en formato .txt suministrados por la entidad certificadora ENAC.

Posteriormente a estos ficheros se les procesan los datos eliminando caracteres especiales y obteniendo todos los campos del .txt, en este procesado a los datos se les realiza un proceso de anonimizado de los datos, el cual consiste en eliminar las columnas con datos personales como matrícula del vehículo, número de bastidor, nombres y apellidos, etc. Mas adelante estos ficheros los transformamos en .csv para poder trabajar más fácilmente con ellos.

Finalmente, terminado el procesamiento de los ficheros obtendremos dependiendo del fichero entre 20000-80000 filas y 16 columnas de las cuales las más interesantes son Inspección, Marca y modelo, CAT, GRUP, INS, Defec.

- Marca y modelo: es el modelo del coche.
- Inspección: es el número único asignado a un coche revisado por un inspector.
- Grupo (GRUP): hace referencia a un conjunto de vehículos (los grupos son más amplios y pueden estar compuestos desde automóviles hasta furgonetas en un mismo grupo).
- Inspector (INS) : es la persona que ha realizado la inspección del vehículo.
- Categoría (CAT.): incluye más tipos de vehículos que el grupo por cada categoría.
- Defectos (Defec.): son los fallos encontrados por el inspector.

Cuando llega un vehículo a una ITV se le asigna un número de inspección único y un inspector, este inspector revisa y apunta los defectos encontrados, si los hay. Por cada defecto encontrado en una inspección se genera una nueva fila con esa inspección.

Es importante señalar que en el proyecto se pueden obtener las reglas de asociación en función de Marca y modelo, Grupo, Inspector y Categoría.

2.4.2. Reglas de asociación en el conjunto de datos del proyecto

Las reglas de asociación (X→Y) en el proyecto se usan para relacionar unos defectos con otros. Para obtener las reglas de asociación fijamos *minsop=0,5*, por consiguiente, obtendremos las reglas cuyo *sop(X)>minsop*, ya que serán los defectos que aparecen más frecuentemente y por lo tanto los más interesantes.

Por otro lado, tenemos la parte derecha de la regla Y, la cual se relaciona con X mediante la confianza. Cuando tenemos una confianza alta entre dos defectos esto tiende a significar que

están relacionados y que cuando se descubre uno de ellos suele aparecer el otro, aunque no siempre se cumple esta premisa.

Un ejemplo de esto puede ser que un inspector siempre encuentre un defecto asociado con las luces traseras junto con otro defecto asociado con el parachoques trasero.

En este caso las luces traseras serían la parte X de la regla y el defecto en el parachoques trasero sería la parte Y de la regla, para denotar la regla se debe hacer como se cita a continuación [luces traseras]→[parachoques trasero]. Sin embargo, esta regla también puede suceder a la inversa [parachoques trasero]→[luces traseras] o puede ser que el soporte de esta regla sea inferior al 0,05 y no sea factible usarla.

Para entender mejor el funcionamiento de las reglas de asociación en este proyecto se va a realizar un ejemplo con las mismas.

En la figura 2.9 podemos ver un ejemplo de 4 reglas de asociación generada a partir de solo los coches que sean de la marca Citroën. Con este filtro se buscarán todas las reglas de los coches que sean de la marca especificada en el filtro.

Citroen

antecedants	consequents	support	confidence	lift	Nº de elementos
['103.0']	['202.0']	0.2413793103448276	0.3842364532019704	1.489137590520079	78.0
['202.0']	['103.0']	0.25802615933412604	0.35944700460829493	1.489137590520079	78.0
['401.0']	['103.0']	0.1700356718192628	0.30769230769230765	1.2747252747252744	44.0
['103.0']	['401.0']	0.2413793103448276	0.2167487684729064	1.2747252747252746	44.0

Figura 2.9: Ejemplo de regla de asociación.

Según el Manual_ITV_V710 [33] los defectos encontrados coinciden con:
103-Placas de matrícula.
202-Carrocería y chasis.
401-Luces de cruce y carretera.

Una de las reglas es [Placas de matrícula]→ [Carrocería y chasis], esta regla crea una relación entre los defectos [Placas de matrículas] y [Carrocería y chasis], la cual puede ser interesante ya que cumple con *sop(X)> minsop*.

El soporte de esta regla es bastante alto ya que en 78 vehículos de un conjunto de aproximadamente 800 se han encontrado defectos lo que supone casi un 10% de los vehículos de ese conjunto.

También es interesante estudiar el valor de la confianza el cual es bastante alto en este ejemplo, este valor nos indica que cada vez que aparece el defecto 103 aparecerá en conjunto con el 202 este hecho se puede interpretar, como que cada vez que encontremos el defecto 103 será bastante frecuente también encontrar el 202.

Esta regla también tiene la inversa de ella la cual se define como [Carrocería y chasis]→ [Placas de matrícula], esto nos permite verificar si también se cumple a la inversa, en este ejemplo sí que se cumplirá que cada vez que se encuentre un defecto en Carrocería y chasis también será probable encontrar un defecto en Placas de matrícula.

Lo explicado anteriormente también se aplica a las reglas [401] → [103] y [103] → [401].
El nombre de los defectos se puede obtener del manual del Ministerio de Industria, Energía y Turismo [33].

3. Localización de anomalías por inspector

En los capítulos anteriores hemos estudiado las reglas de asociación aplicadas al caso de las auditorías a las inspecciones técnicas de vehículos, pero siguiendo una metodología estándar; en realidad se han detectado reglas sobre un conjunto de ítems y transacciones, sin que el hecho de que estos ítems fueran defectos encontrados y las transacciones todos los defectos para un vehículo concreto supusieron ninguna particularidad.

El aspecto más novedoso del trabajo surge porque las transacciones dependen de un factor adicional: la persona que ha realizado la inspección. En efecto, diferentes inspectores pueden tener tendencia a considerar distintas reglas de asociación de manera implícita. Por ejemplo, un inspector puede estar habituado a que siempre que encuentra un defecto se encuentre asociado a otro defecto, y esa "costumbre" le llevará a buscar el defecto asociado más que en el caso de otros compañeros.

Una de las motivaciones de este trabajo es tratar de detectar este tipo de situaciones.

3.1. Definición de anomalía en la inspección

Partimos de un conjunto de reglas de asociación R, todas con un soporte mínimo de 0.05 y de un conjunto de inspectores I. Vamos a definir dos tipos de anomalías, una relacionada con diferencias en el soporte y otra asociada a diferencias en la confianza de la regla.

Definición 1

- Decimos que un inspector $i \in I$ ha cometido una anomalía de soporte para una regla de asociación dada $X \rightarrow Y \in R$, si la cantidad media de inspecciones en las que i ha detectado el conjunto de defectos X es significativamente distinta al número medio de inspecciones en las que el resto de los inspectores han encontrado estos mismos defectos.

- Decimos que un inspector $i \in I$ ha cometido una *anomalía de confianza* para una regla de asociación dada $X \rightarrow Y \in R$, si la cantidad media de inspecciones en las que, habiendo detectado los defectos X, se encuentran también los defectos Y, es significativamente diferente a la misma medida, pero obtenida considerando el resto de los inspectores.

Observación 1

Hay que observar que ambas anomalías son independientes. Por ejemplo, un inspector puede tener tendencia a encontrar muchos más defectos en las luces traseras que sus compañeros, pero en cambio, de todas las inspecciones que detecta con defecto en luces traseras, encuentra, por ejemplo, la misma proporción de matrículas erróneas que sus compañeros. Esto sería una anomalía de soporte, pero no de confianza.

Observación 2

Las anomalías pueden ser negativas o positivas, según la diferencia de las medias. Podemos hablar de anomalías de soporte negativa para el inspector i.

Aunque un inspector no tenga anomalías de estos tipos puede suceder que presente diferencias al compararse con un compañero particular. Esta situación se define en la siguiente sección.

3.2. Definición de conflicto en la inspección

Dentro de las posibles discrepancias entre dos inspectores concretos, a las que llamaremos conflictos, podemos distinguir dos casos: uno asociado al soporte y otro asociado a la confianza de la regla considerada.

Definición 2

- Dos inspectores $i_1, i_2 \in I$ presentan un *conflicto de soporte* cuando para una regla de asociación dada $X \rightarrow Y \in R$, la cantidad media de inspecciones en las que i_1 ha detectado (al menos) los defectos X, es significativamente distinta a la media de veces que han encontrado estos defectos i_2.

- Dos inspectores $i_1, i_2 \in I$ presentan un conflicto de confianza cuando para una regla de asociación dada $X \rightarrow Y \in R$, la cantidad media de inspecciones en las que i_1, habiendo detectado los defectos X encuentra también los defectos Y, es significativamente diferente a la de i_2.

Los conflictos también cumplen las observaciones 1 y 2.

Observación 3

Si el conjunto de datos solo incluye dos inspectores, un conflicto es también una anomalía y viceversa. Esto ha sido útil en la implementación, al permitirnos definir las anomalías como un "conflicto" entre un inspector y el "inspector resto" que agrupa a las demás inspectores, de forma que solo ha habido que implementar la noción de conflicto.

4. Métodos para detectar anomalías

Para detectar anomalías consideramos por un lado a cada inspector *i* junto con sus inspecciones, y 'marcamos' el resto de las inspecciones como si hubieran sido realizadas por un inspector imaginario i_2. Según las definiciones 3.1 y 3.2, encontrar un conflicto en esta situación será lo mismo que haber encontrado una anomalía.

Estas definiciones se basan en encontrar diferencias significativas de ciertos valores promedios. Con el fin de detectar si, en efecto, las diferencias entre medias resultan significativas, hemos estudiado y llevado a cabo diversas pruebas con los test estadísticos que se explicaran a continuación.

4.1. Conceptos

En este apartado se hará referencia a conceptos de la estadística que posteriormente se necesitan para entender los test explicados en el siguiente epígrafe.

4.1.1. Test estadístico

Un test estadístico [29, 10] es una prueba matemática que sirve para validar o rechazar las hipótesis de modelación probabilistas. Los test tratan de distinguir lo que es verosímil de lo que es muy poco verosímil, en el marco de un modelo dado.

Estos test intentan determinar por ejemplo si dos muestras corresponden a la misma población. En este sentido, cuanto mayor sean las muestras más fácil será ver si las diferencias son estadísticamente significativas.

Por ejemplo, supongamos que estamos analizando dos secuencias de tiradas de monedas, y queremos averiguar si se trata de dos secuencias de tiradas diferentes de la misma moneda (o de dos monedas con las mismas características), o si se trata de secuencias de dos monedas diferentes.

Si cada secuencia consta de solo 2 tiradas, aunque sean resultados completamente diferentes, por ejemplo, CC y XX, el test nos dirá que no es suficiente para asegurar que se trata de monedas diferentes. Pero si las secuencias son de 1000 tiradas y la primera es C…C y la segunda X…X, el test nos dirá que es sumamente improbable que una misma moneda pueda generar ambas.

4.1.1.1. Contraste de hipótesis

El contraste de hipótesis [6] es un procedimiento para juzgar si una propiedad que se supone en una población estadística es compatible con lo observado en una muestra de dicha población.

Existen dos tipos de hipótesis; La *hipótesis nula* [35] se denota como H_0,es la expresión que se quiere verificar en el test. Los test están diseñados para verificar la fuerza de la evidencia en contra de la hipótesis nula. Por otro lado, tenemos la *hipótesis alternativa* [9] la cual se denota como H_1.

Cuando la *hipótesis nula* no se cumple decimos que se verifica la *hipótesis alternativa*. En el ejemplo de las monedas del apartado anterior la hipótesis nula se enuncia como ambas secuencias de tiradas corresponden a monedas con las mismas características.

A continuación, vamos a citar otro ejemplo del contraste de hipótesis [9].

Tras la creación de la hipótesis "El alcohol te hace caer" se crean dos hipótesis.
Hipótesis alternativa: las personas que beben alcohol se caerán más que aquellos que no beben alcohol.
Hipótesis nula: las personas se caerán la misma cantidad de veces independientemente de si han bebido alcohol.

Normalmente la hipótesis nula nos indica que no hay relación entre dos fenómenos, como en el caso anterior entre el alcohol y las caídas.

El propósito principal de los test estadísticos es determinar si se puede descartar la hipótesis nula por ser muy improbable. Para esto se utiliza un nivel se significancia p. Por ejemplo, se puede fijar p = 0.001, indicando que rechazaremos la hipótesis nula (por poco probable) si el test garantiza que no se verificará más de una de cada mil veces.

4.1.1.2. Test paramétricos

Las pruebas paramétricas [34] asumen distribuciones estadísticas subyacentes a los datos. Por tanto, deben cumplirse algunas condiciones de validez, de modo que el resultado de la prueba paramétrica sea fiable.

Las condiciones requeridas por estos test, por ejemplo, seguir cierta distribución de probabilidad, hace que sea más fácil obtener resultados significativos con tamaños de muestra menor que en el caso de los test no paramétricos.

4.1.1.3. Test no paramétricos

Las pruebas no paramétricas [34] no necesitan ajustarse a ninguna distribución. Pueden por tanto aplicarse incluso aunque no se cumplan las condiciones de validez paramétricas. Son menos robustas que las paramétricas, pero son válidas para un espectro más amplio.

4.1.1.4. p-value

p-value [37, 38] sirve para determinar cuál es el significado obtenido de un test estadístico, en función del valor de p-value se validará la hipótesis alternativa o la hipótesis nula. Cuanto menor sea el valor de p-value más fuerte será la evidencia en favor de la hipótesis alternativa, en contraposición cuanto mayor sea p-value más fuerte es la evidencia a favor de la hipótesis nula.

4.1.1.5. t-statistics

t-statistics [38] o t es un valor que va en conjunto con p-value, es el cálculo de la diferencia en unidades de error estándar. Cuanto mayor es la magnitud t más posibilidades hay de que se acepte la hipótesis nula ya que es más probable que haya una diferencia significativa, sin embargo, si t tiende a 0 es más probable que no haya diferencia significativa.

4.1.1.6. Pruebas de colas

Las pruebas de colas [36, 9] son un tipo de test estadístico que se usan en función del potencial diferencial de las medias obtenidas en nuestra muestra.

P**rueba de dos colas o bilateral:** Se asocia a una hipótesis alternativa para la cual se desconoce el potencial diferencia de medias. Se usa para comparar dos medias en las cuales no se sabe cuál es mayor de las dos. Por ello se ha de verificar en ambos sentidos de la prueba [36, 9].
media(A) ≠ media(B).

P**rueba de una cola o unilateral:** Se asociada a una hipótesis alternativa para la cual se conoce el signo de la potencial diferencial antes de ejecutar el experimento y la prueba.
En función de que la **media(A) < media(B)** o **media(A) > media(B)** la dirección esperada de la diferencia tendrá un sentido u otro [36, 9].

4.2. Test Estadísticos estudiados

En este apartado se comentarán los test que se han estudiado para implementarlos en el proyecto.

El procedimiento para aplicar un test es en general como el que se muestra a continuación [9].

1. Introducir los datos.
2. Explorar los datos: Es interesante realizar un análisis exhaustivo de los datos, realizando gráficos y estadística descriptivas.
3. Realizar los test: En función de los resultados obtenidos previamente es posible que sea necesario lanzar test más robustos.
4. Cálculo de los efectos: sirve para cuantificar su efecto.

4.2.1. Test de la t-Student

La t -Student [10, 30] o t-test, es un test que desarrolló William Sealy Gosset [28] el cual adoptó el seudónimo Student para realizar diversas publicaciones. Este test lo desarrolló mientras estudiaba las mejores variedades de cebada en las destilerías Guinness.

El t-test inicialmente se diseñó para examinar las diferencias entre las medias de dos muestras independientes que tengan distribución normal y homogeneidad en sus varianzas.

El test de la t-Student se fundamenta en dos premisas;

- Las muestras deben tener una distribución normal.
- Las muestras deben ser independientes. Esta premisa se refiere a que un mismo elemento de la muestra no puede estar contenido en las dos muestras que se comparan.

Es importante destacar que, aunque la t-Student cuente con el requisito de que las muestras deben tener una distribución normal, para poblaciones grandes este factor deja de ser relevante [45].

Se debe tener en cuenta que en el t-test una desviación estándar es la diferencia estándar de error. Un error estándar en la mayoría de las parejas de muestras de una población significa que ambas muestras tienen una media muy similar o una diferencia pequeña, sin embargo, dos muestras de población con un error estándar muy grande hacen alusión que las medias difieren en gran medida unas de otras.

El t-Test puede ser de dos tipos independiente o dependiente en función de unas condiciones u otras de las muestras se podrá usar uno u otro. A continuación, se explicarán las condiciones y características de los t-test independientes y los t-Test dependientes:

4.2.1.1. Dependiente

El t-Test Dependiente [9, 10] se usa para comparar las medias de dos grupos que incluyen los mismos participantes, aunque bajo diferentes condiciones. Obviamente, los grupos deben ser de igual tamaño.

Un ejemplo de uso del t-Test dependiente podría realizarse sobre un grupo de 10 personas en el que queremos medir si correr 10 km dispara las pulsaciones para ello mediremos las pulsaciones del grupo de participantes antes de correr los 10 km y después de correr.

Hipótesis Nula: Las medias son iguales antes y después de correr.
Hipótesis Alternativa: Las medias obtenidas son diferentes y las pulsaciones son mayores después de correr que antes de correr.

Para poder realizar el t-Test dependiente primero se debe calcular los valores que se muestran en la tabla 4.1.

Variable	Observaciones	Mínimo	Máximo	Media	Desv. típica	Error Std.
Antes de 10km	21	33,000	68,000	53,571	11,435	2,49
Después 10km	21	70,000	144,000	113,190	24,005	4,58

Tabla 4.1: Tabla t-Test dependiente.

Los valores que vamos al obtener después de realizar todas las operaciones para calcular la función de distribución son el valor de t, los grados de libertad y valor de p-value el cual nos indica si hay una gran diferencia entre las medias calculadas, tal como se muestra en la tabla 4.2.

Diferencia	-59,619
t (Valor observado)	-10,275
|t| (Valor crítico)	2,021
GL	40
valor-p (bilateral)	< 0,0001
alfa	0,05

Tabla 4.2: Tabla t-Test dependiente p-value.

Si el valor de p-value es <0.05 se rechaza la hipótesis nula dando por válida la hipótesis alternativa verificando que si cambian las pulsaciones después de correr 10 km.

Este ejemplo está inspirado en un ejemplo del libro [10, 39].

4.2.1.2. Independiente

El t-Test Independiente [9, 10] se utiliza para comparar las medias de dos grupos integrados por diferentes participantes, respetando la condición de que un mismo elemento sólo debe participar una vez en uno de los grupos. En esta ocasión los grupos pueden ser de diferente tamaño.

A continuación, citaremos un ejemplo para que quede más claro cómo funciona el t-test independiente.

Para realizar un ejemplo de t-Test Independiente vamos a coger un conjunto de 10 personas que leyeron IT y otro grupo de 10 personas que leyeron la Llamada de Cthulhu y la variable objetiva es el terror sentido después de leerlos libros. Para valorar el nivel de este miedo, se les pidió a los participantes que le dieran una puntuación entre 0-100 siendo 0 que no da miedo y 100 mucho miedo.

Hipótesis nula: La media de terror sentido después de leer el libro será igual en los dos grupos tras leer los libros.

Hipótesis alternativa: Las medias de terror sentido tras leer los libros son diferentes una respecto a la otra.

Las personas que leyeron IT sienten un terror con una puntuación media de 20 con una desviación estándar de 4,11, sin embargo, el grupo que leyó La llamada de Cthulhu siente un terror con una puntuación media de 24 con una desviación estándar 4,70. Los resultados se muestran en la tabla 4.3.

Variable	Observaciones	Mínimo	Máximo	Media	Desv. típica	Error Std.
Lectores IT	10	-	-	20,0	4,10	1,29
Lectores La llamada de Cthulhu	10	-	-	24,0	4,70	1.48

Tabla 4.3: Tabla t-Test independiente.

Por último, calculamos los grados de libertad.
Mediante el cálculo de la función de distribución obtenemos el valor de p-value=0.048 el cual en este caso es igual para ambos valores de la prueba de dos colas.

Puesto que p-value < 0,05 podemos concluir que hay una diferencia significativa entre las dos medias, por lo tanto, se rechaza la hipótesis nula dando por válida la hipótesis alternativa, como se muestra en la tabla 4.4.

G	18 17,678
t (Valor observado)	-2,125 -2,125
p-value	0,048 0,048

Tabla 4.4: Tabla t-Test Independiente.

Observando las medias podemos concluir que el miedo sufrido después de leer la Llamada de Cthulhu es mayor que el miedo que mostraron las personas que leyeron IT.

Este ejemplo está inspirado en un ejemplo del libro [10].

Un ejemplo del cálculo de t-test independiente en Python sería como el que se muestra a continuación. Tenemos dos conjuntos de datos rvs1 y rvs2 los cuales tienen un gran número de 1 y 0, además estos conjuntos para hacerlos más grandes los vamos a extender con una gran cantidad de 0.

Una vez extendidos usamos la función ttest_Ind la cual nos devolverá un valor de statistic y p-value figura 4.1, el p-value obtenido nos indica que la diferencia entre los dos conjuntos es estadísticamente significativa.

```
from scipy import stats
rvs1=[1,0,0,0,1,1,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0,0,0,1,
1,0,0,0,1,0,0,0,1,0,0,1,1,1,1,1]
rvs1.extend([0]*10000)
rvs2=[0,1,0,0,0,1,0,0,0,1,0,0,1,0,0,0,0]
rvs2.extend([0]*1000000)
t=stats.ttest_ind(rvs1,rvs2,equal_var=False)

Ttest_indResult(statistic=3.992887434778269,pvalue=6.5738412777277067e-05)
```

Figura 4.1: Código t-Test independiente.

p-value es el valor más interesante de los dos ya que nos permitirá saber cuánto difieren las dos medias. Si el valor de p-value < 0.05 se puede asegurar que las medias son significativamente diferentes la una de la otra. Otro factor a tener en cuenta es que cuanto más pequeño sea el valor de p-value más difieren las medias calculadas.

4.2.2. Kolmogorov-Smirnov

El test de Kolmogorov-Smirnov es un test estadístico desarrollado por los matemáticos rusos Andrey Kolmogorov and Nikolai Smirnov [27, 40, 10, 41].

El test de Kolmogorov-Smirnov o test K-S es un test no paramétrico de igualdad para distribuciones de probabilidad unidimensionales continuas que se usan para comparar dos muestras.

Es importante destacar que el test de Kolmogorov-Smirnov es un test que depende en gran medida del tamaño de los conjuntos de datos usados, siendo este significativamente relevante para los resultados que posteriormente nos producirá.

Como ya se ha comentado en el t-Test el test de Kolmogorov-Smirnov también cuenta con una hipótesis nula y una hipótesis alternativa, las cuales en función del valor de p-value se validará la hipótesis nula o la hipótesis alternativa.

Cuando p-value >0,05 el test la distribución de la muestra no será significativamente diferente y por lo tanto se rechaza la hipótesis alternativa. Sin embargo, sí p-value <0,05 la distribución será diferente de una distribución normal y se rechazará la hipótesis nula.

Veamos un ejemplo de aplicación del test de Kolmogorov-Smirnov.

Tenemos dos conjuntos de personas un conjunto de 1000 y otro de 500 a las cuales después de haberse leído el Quijote se las ha pedido que valoren de 0-100 cómo de aburrido les ha parecido.

Queremos averiguar si las muestras pertenecen al mismo conjunto de datos o son independientes.

hipótesis nula: La distribución de las muestras es la misma.
hipótesis alternativa: Las distribuciones de las muestras son diferentes.

De estos conjuntos hemos calculado la media máximo y mínimo y la desviación típica de los conjuntos tal y como se muestra en la tabla 4.5.

Variable	Observaciones	Mínimo	Máximo	Media	Desv. típica	Error std.
Aburridometro (Muestra 1)	1000	0,000	100,000	51,371	28,735	0,90
Aburridometro (Muestra 2)	500	0,000	100,000	24,641	15,190	0,60

Tabla 4.5: Tabla test Kolmogorov-Smirnov.

Realizando las operaciones para calcular las funciones de distribución de las muestras obtenemos los valores que se muestran en la siguiente tabla 4.6.

D	0,500
valor-p (bilateral)	< 0,0001
alfa	0,05

Tabla 4.6: Tabla test Kolmogorov-Smirnov p-value.

Puesto que el resultado obtenido p-value <0,0001 podemos decir que se rechaza la hipótesis nula por lo tanto aceptamos la hipótesis alternativa y llegamos a la conclusión de que las muestras son independientes.

Un ejemplo del cálculo del test K-S en Python sería como el que se muestra a continuación. Tenemos dos conjuntos de datos rvs1 y rvs2 los cuales tienen un gran número de 1 y 0 ,además estos conjuntos para hacerlos más grandes los vamos a extender con algunos 0. Una vez extendidos usaremos la función `ks_2samp` la cual nos devolverá un valor de statistic y p-value.

Si p-value<0,05 podemos asegurar que la diferencia entre las medias es estadísticamente significativa como se muestra en la figura 4.2.

```
from scipy import stats
rvs1=[1,0,0,0,1,1,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0,0,0,0,1,0,0,0,1,
1,0,0,0,1,0,0,0,1,0,0,1,1,1,1,1]
rvs1.extend([0]*20)
rvs2=[0,1,0,0,0,1,0,0,0,1,0,0,1,0,0,0,0]
rvs2.extend([0]*150)
print(stats.ks_2samp(rvs1,rvs2))

Ks_2sampResult(statistic=0.22220175034546297, pvalue=0.016362234929021557)
```

Figura 4.2: Código Kolmogorov-Smirnov.

4.2.3. Wilcoxon

El test de Wilcoxon [24, 10] es un test no paramétrico que permite comparar poblaciones cuando sus distribuciones no satisfacen las condiciones necesarias para otros test paramétricos.

Se usa como alternativa al t-test de muestras dependientes cuando las muestras no siguen una distribución normal o el tamaño es demasiado pequeño para determinar si las muestras proceden de poblaciones normales.

El test de Wilcoxon tiene las características que se citan a continuación:

- Los datos tienen que ser dependientes, es decir los datos de una muestra debe estar relacionados con los datos de la muestra con la que se va a comparar.
- Tener la capacidad de ordenar de menor a mayor o viceversa.
- El tipo de distribución de las diferencias tiene que ser simétrica.

- El test de Wilkinson trabaja con medianas.
- Es mejor opción que t-Test cuando hay valores atípicos, no hay normalidad de los datos o el tamaño de las muestras es pequeño.

4.2.4. Explicaciones de uso de los test estadísticos

Una vez analizados y explicados los test estadísticos que creemos que pueden ser interesantes para el proyecto pasaremos a explicar por qué se han usado o descartado y qué usos se les podría dar a dichos test estadísticos.

Primero debemos definir la variable aleatoria que se considera para realizar los test estadísticos. Dado un inspector i, un conjunto de defectos X, una secuencia de inspecciones S, se considera la variable aleatoria, que se obtiene al asignar a cada elemento de S el número de veces que i ha encontrado X en S. Como un mismo efecto solo se puede encontrar una vez (o ninguna), la variable tiene el aspecto de variable binaria o categórica (a las que se suele llamar dummy en estadística).

Los dos test que se han usado en este proyecto han sido el t-Student independiente y el test de Kolmogorov-Smirnov.

El t-test de independiente es el que hemos usado en primera instancia para detectar en cuánto difieren las medias de dos conjuntos de inspectores y poder saber si existe una anomalía con un inspector o un conflicto de inspectores. Normalmente si dos inspectores tienen un conflicto en el que difieren sus medias en gran medida también habrá una anomalía.

Sin embargo, nos percatamos que debido al proceso de anonimización de los datos ya no podíamos distinguir si un mismo vehículo había pasado por dos inspecciones distintas en la misma ITV, es decir si aparecía en dos muestras diferentes el mismo vehículo.

Por ello para realizar una aplicación más fiable decidimos incluir también el test de Kolmogorov-Smirnov en el cual no es relevante si un mismo vehículo aparece en dos muestras

diferentes. Puesto que es un test no paramétrico y puede abarcar un espectro mayor, pero necesita muestras de datos mayores.

Por otro lado, tenemos t-Test dependiente, el cual nos hemos visto obligados a descartar ya que no cumplimos con las condiciones necesarias para poder usarlo. Nuestros conjuntos de inspectores no son del mismo tamaño y no se puede saber si un mismo coche ha sido revisado por inspectores diferentes.

En última instancia, el test de Wilcoxon no es aplicable sobre todos los datos del conjunto, puesto que una de las condiciones es que deben ser el mismo número de elementos en las dos muestras.

Aunque el t-test de dependiente y el test de Wilcoxon no lo hemos usado, consideramos que podría ser interesante mencionarlos en esta memoria, ya que podría ser útil para trabajos futuros como ver la evolución en la detección de defectos.

4.2.5. Resultados: anomalías detectadas

En este apartado se explicarán los resultados obtenidos de aplicar los diversos test para comparar los defectos detectados por los inspectores.

Ejemplo 1

Para la primera prueba de concepto vamos a coger un resultado de analizar el fichero "PITLR5.csv" para el conjunto de todos los coches del dataset.

Obtenemos como resultado de la aplicación la comparación entre el inspector 436 con el resto de los inspectores usando la regla 901 → 103 como referencia en las imágenes se muestra que hay una anomalía de soporte en cuanto a la parte izquierda de la regla en concreto en el defecto 901, ya que el inspector 436 detecta una notable mayor cantidad de defectos 901 que el resto de los inspectores, como se muestra en la figura 4.3.

Inspector1(NTotalCoches/X(RatioX)/Y(RatioY))	Inspector2(NTotalCoches/X(RatioX)/Y(RatioY))
all(7930/2077(3.82)/698(2.98))	436.0(1116/382(2.92)/100(3.82))

ReglaX	ReglaY	XT-Student	X-Kolmogorov-Smirnov
901.0	103.0	1.065159455570476e-07	5.857871884477615e-06

YT-Student	Y-Kolmogorov-Smirnov
0.002857163732981355	0.05405624828781196

Figura 4.3: Ejemplo 1 test estadísticos.

Inspector 436, ha realizado 1116 inspecciones, y en 382 de ellas encuentra el defecto 901, es decir 1116/382=2,92 o aproximadamente en 1 de cada 3 inspecciones.

El total de los inspectores ha realizado 7930 inspecciones, y en 2077 se encuentra el defecto 901, es decir de media 7930/2077=3,82, es decir una de cada 4 inspecciones.

En el caso de la t-Student se muestra que la diferencia de medias es significativa y en el caso del test de Kolmogorov-Smirnov se verifica que las muestras corresponden a distintas distribuciones de probabilidad; esto solo pasaría al azar alrededor de 1 caso por millón de inspectores considerados, estas conclusiones se pueden deducir de valor de p-value.

En este caso solo se produce qué es la parte izquierda de la regla la que genera la anomalía, pero se puede dar el caso que sea solo la parte derecha de la regla (En el ejemplo el defecto 103) o ambas partes.

Ejemplo 2

En el segundo ejemplo vamos a coger un resultado de analizar el fichero “PITLR5.csv” para el conjunto de todos los coches del dataset.

Hemos obtenido como resultado de la aplicación que el inspector 430 comparado con el resto

respecto a la regla 103→202 detecta menos defectos 103 que el resto de los inspectores por lo tanto encontramos una anomalía de soporte.

Por otro lado, el inspector 430 también tiene una anomalía de confianza con respecto al resto de los inspectores ya que detecta un mayor número de defectos que el resto de los inspectores de el defecto 202 siempre que ha detectado el defecto 103 previamente, esto se ve reflejado en la figura 4.4.

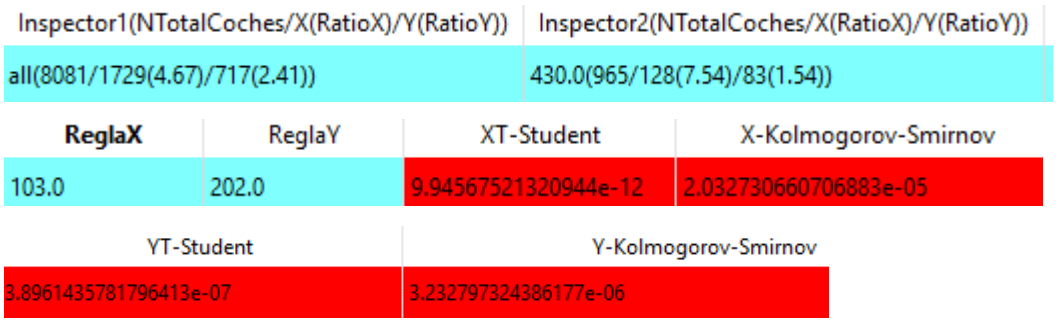


Figura 4.4: Ejemplo 2 test estadísticos.

Inspector 430, ha realizado 965 inspecciones, y en 128 de ellas encuentra el defecto 103, es decir 965 /128 =7,53, o aproximadamente en 1 de cada 7 inspecciones

El total de los inspectores ha realizado 8081 inspecciones, y en 1729 se encuentra el defecto 103 de media 8081/1729=4,67, es decir una de cada 5 inspecciones.

El inspector 430 también tiene una anomalía de confianza la cual es conveniente analizar puesto que relaciona los dos defectos mediante la regla 103→202. El inspector 430 de esos 128 casos en los que encuentra el defecto 103 también encuentra el defecto 202, 83 veces esto ocurre 128/83=1,54, aproximadamente esto ocurre más de la mitad inspecciones.

El total de inspectores con respecto a la regla 103→202 han detectado el defecto 202 en 717 vehículos de los 1729 en los que encuentran el defecto 103 esto ocurre 1729/717=2.41, aproximadamente esto ocurre en 1 de cada 3 inspecciones.

En este caso se cumple lo mismo que en el ejemplo1 para t-Student y para el test de Kolmogorov-Smirnov. Lo que le hace diferente a este ejemplo es que además de cumplirse en la parte izquierda de la regla (X) también se cumple en la parte derecha (Y). Esto supone que hay una diferencia significativa en la cantidad de veces que detecta estos defectos juntos respecto a otros inspectores.

Ejemplo 3

Por último, se va a mostrar un ejemplo de conflicto entre dos inspectores 436 y 430 en relación a la regla 901→202 obtenido del fichero "PITLR5.csv".

Se ha detectado que existe un conflicto de soporte y de confianza mediante los test de t-Student y Kolmogorov-Smirnov, figura 4.5.

Inspector1(NTotalCoches/X(RatioX)/Y(RatioY))	Inspector2(NTotalCoches/X(RatioX)/Y(RatioY))
436.0(1116/382(2.92)/129(2.96))	430.0(965/154(6.27)/81(1.9))

ReglaX	ReglaY	XT-Student	X-Kolmogorov-Smirnov
901.0	202.0	1.4057080934024005e-22	1.3501399696365232e-15

YT-Student	Y-Kolmogorov-Smirnov
8.223028874836082e-05	0.0006870364397147394

Figura 4.5: Ejemplo 3 test estadísticos.

En relación al conflicto de soporte el inspector 436 detecta 1116/282=2,92 aproximadamente 1 de cada 3 inspecciones el defecto 901, sin embargo, el inspector 430 lo detecta 965/154 aproximadamente 1 de cada 6 inspecciones esto supone una diferencia sustancial en cuanto a la detección del defecto 901 generando un conflicto de soporte.

Por otro lado, tenemos el conflicto de confianza en el que el inspector 436 obtiene 382/129=2,96 de las 382 veces que detecta el defecto 901, 129 veces ha detectado también el defecto 202 esto supone aproximadamente 1 de cada 3 inspecciones.

En cuanto al inspector 430 a detectado 154/81=1.9 lo que supone que 1 de cada 2 inspecciones en las que detecta el defecto 901 también detecta el defecto 202.

5. Tecnologías utilizadas en la aplicación

Este capítulo detalla cuáles son las tecnologías usadas para el desarrollo de la aplicación, exponiendo además porque son las que más nos conviene para el desarrollo de este proyecto.

5.1. Sublime-text 3 , Python 3.7

Como entorno de desarrollo he usado el editor de texto Sublime-text [20], que con los plugin adecuados puede convertirse en un entorno de desarrollo amigable y ágil. También admite gran cantidad de lenguajes de programación como Python, C, C++, Java …

Alguno de los plugins más útiles que usado para desarrollar la aplicación han sido:
Anaconda: Intérprete de Python.
SidebarEnhancements: Añade un árbol con los archivos y carpetas.
Alignment: Permite alinear el código.
Git Gutter: Control de versiones permite ver los cambios que se han hecho en el momento.

En un principio se barajó la posibilidad de hacer el desarrollo de la aplicación en R, finalmente por motivos de versatilidad se decidió realizar el desarrollo de la aplicación en Python, R está diseñado en exclusiva para el análisis de datos siendo Python más versátil, por ejemplo, para realizar una aplicación con interfaz gráfica.

5.2. PyQt5

PyQt5 [16] es una biblioteca que sirve de enlace para conectar Python con Qt5 el cual está formado por una gran cantidad de bibliotecas desarrolladas en C++.

Estas bibliotecas implementan APIs que permiten acceder a muchos aspectos de los desktop y móviles modernos. También incluyen algunos servicios como el de ubicación y posicionamiento, multimedia, Bluetooth, …

Una de las cualidades de PyQt5 es que es multiplataforma ya que soporta gran cantidad de SO como Windows, Linux, UNIX, Android, OS X y plataformas de iOS.

Otro factor determinante para elegir PyQt5 frente a frente a otras bibliotecas similares como WxPython, PySide o PyGTK es la amplia comunidad que tiene facilitando encontrar respuestas a los múltiples problemas a la hora de generar la interfaz de la aplicación.

5.3. Matplotlib

Matplotlib [12] es una biblioteca desarrollada para Python que permite generar figuras en 2D y 3D para una gran variedad de plataformas y formatos. Algunas de las figuras que puede generar son gráficos, histogramas, espectros de potencia, diagramas de errores, diagramas de dispersión, etc. Un ejemplo de figura que puede generar esta biblioteca es la que se muestra en la figura 5.1.

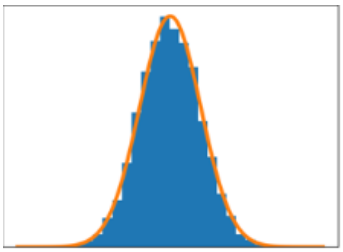


Figura 5.1:Matplotlib ejemplo

En la aplicación esta biblioteca se ha usado para generar un histograma que muestre cuántos vehículos por número de defectos en función de la categoría del vehículo, el grupo al que pertenece, por inspectores, por marca y modelo. Esta funcionalidad de la aplicación se tratará más en detalle en el próximo capítulo.

5.4. Statistics

Statistics [18] es una biblioteca desarrollada en Python que permite realizar operaciones de cálculo estadístico. Está estructurada en dos grupos un grupo se utiliza principalmente para cálculo de medias , promedios, etc y otro grupo que se utiliza para calcular las dispersiones de los datos de un conjunto de datos determinado.

El cálculo de la media, mediana y moda pertenecen al primer grupo y el cálculo de la desviación típica y varianza pertenecen al segundo grupo.

Un ejemplo de funcionamiento sería el que se muestra en la figura 5.2.

```
import statistics as stats
edades=[10,12,14,16,20,22,24,24,26,28]
print(stats.mean(edades))
19.6
print(stats.pvariance(edades))
35.04
```

Figura 5.2: Ejemplo librería statistics.

5.5. Scipy

Scipy [13] es una colección de algoritmos matemáticos y funciones creadas como extensión de la biblioteca Numpy para Python. Permite manipular y visualizar grandes cantidades de datos.

Gracias al Scipy Python puede dar soporte al desarrollador para crear aplicaciones sofisticadas y específicas para ambientes más científicos.

En el proyecto el módulo que más hemos usado es scipy.stats el cual permite calcular distribuciones continuas, distribuciones discretas, t-Test y KS-test. Se han usado ambos módulos el t-Test para realizar el test estadístico de la t-Student (ttest_ind) y él KS-test para calcular el test de Kolmogorov-Smirnov (ks_2samp).

5.6. Mlxtend

Mlxtend [22] es una biblioteca desarrollada para Python, que contiene multitud de herramientas y algoritmos desarrollados principalmente para el aprendizaje automático, y en general para la ciencia de datos

El objetivo de esta biblioteca según sus creadores es dar acceso a los algoritmos de machine learning a investigadores y trabajadores de este campo. Mediante una API sencilla e intuitiva y con gran compatibilidad con otras librerías de machine learning.

Esta biblioteca engloba módulos de regresión ,preprocesamiento de clúster(k-means), clasificadores, patrones frecuentes , dibujo de figuras, etc.

En el proyecto concretamente se han usado los módulos de cálculos de reglas de asociación y del algoritmo de Apriori los cuales pertenecen al módulo de patrones frecuentes.

5.7. Pandas

Pandas [17, 21] es una biblioteca con licencia BSD desarrollada en Python como extensión de Numpy facilita el análisis de datos proveyendo de estructuras de datos y herramientas para el análisis de datos.

Algunas de las características de esta librería es la capacidad de obtener datos de numerosas fuentes como ficheros de texto, bases de datos y archivos Excel, también permite la organización de los datos realizando uniones, inserción y eliminación de datos, etc.

Con respecto a la aplicación esta librería se ha usado para la obtención de los datos de los Excel, para generar ficheros csv y para el acceso a los datos con el propósito de realizar los cálculos requeridos en la aplicación.

5.8. Py2exe

Py2exe [1] es una utilidad de Python que permite generar ejecutables en Windows a partir de las librerías de los scripts de la aplicación. Su principal cualidad es que permite ejecutar el script sin tener Python instalado.

5.9. Control de versiones

GitHub me ha resultado muy útil para poder agilizar el proceso de descargar el código y ponerlo en funcionamiento. También me ha permitido volver a versiones anteriores cuando algún cambio realizado había salido mal.

6. Aplicación

En este capítulo se explicará detalladamente las funcionalidad, y algunas de las vistas principales de la aplicación.

El objetivo de la aplicación es analizar los datos obtenidos de las ITV de los vehículos pudiendo obtener métricas que ayuden a detectar anomalías en las inspecciones que puedan ser justificadas como estadísticamente muy improbables. Con este objetivo se han implementado varias funcionalidades que darán solución a este problema y facilitarán su análisis.

6.1. Funcionalidades

En este apartado explicaremos las funcionalidades principales de la aplicación, explicando en detalle las acciones que puede realizar el usuario.

6.1.1. Cargar archivo

Los datos se obtuvieron en primera instancia desde ficheros de texto en los que hubo que hacer un tratamiento de datos muy exhaustivo de los ficheros, ya que estos contaban con unas cabeceras que se repetían varias veces durante el fichero y la única forma de detectar a qué grupo de la cabecera pertenece cada dato era a través de unos guiones situados justo después de la cabecera.

A esto se le suma que cada fichero tenía algunos campos que difieren de los otros ficheros haciendo que fuese casi obligatorio revisar que la obtención de los datos fuera correcta.

También fue necesario un proceso de anonimización de los ficheros para eliminar datos personales y cumplir con la RGPD(Reglamento General de Protección de Datos).

En definitiva, la obtención de los datos desde los ficheros de texto fue un trabajo arduo y laborioso.

Además, se puede realizar la obtención de los datos mediante ficheros Excel y ficheros csv. Esta opción es más aconsejable, ya que conlleva menos errores en cuanto formato puesto que se usa pandas para su lectura, haciendo alusión a la su eficiencia es más rápido que la obtención mediante fichero. La carga del fichero se muestra en la figura 6.1.

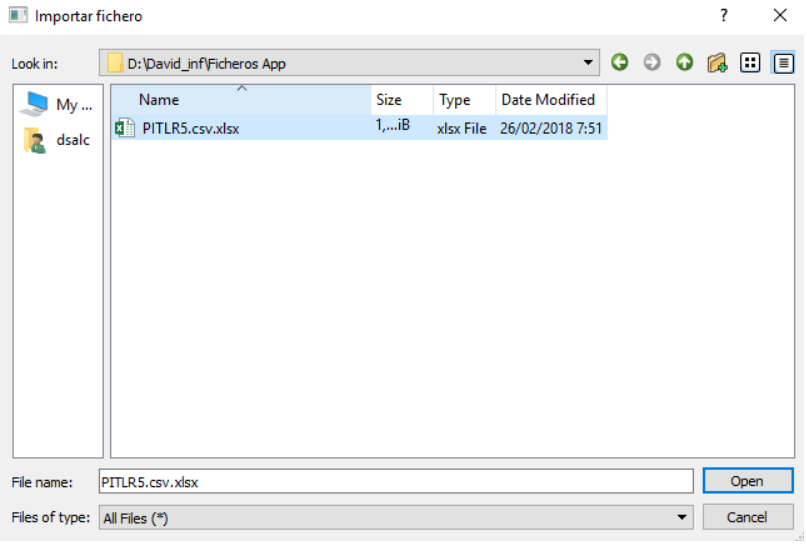


Figura 6.1: Carga de ficheros.

6.1.2. Filtrar Datos

Para facilitar la gestión de los datos cargados desde los archivos, la aplicación permite filtrarlos seleccionado en la cabecera del campo que se quiera filtrar. Esta acción desplegará un menú como se muestra en la figura 6.2.

En el menú desplegado se puede:
Cancelar la acción, limpiar los filtros, eliminar todos los filtros dejando la tabla como estaba en el origen, seleccionar todos los datos y por último aceptar cuando se quieran guardar los cambios en el filtro, está acción producirá que se muestren únicamente los elementos seleccionados en el menú.

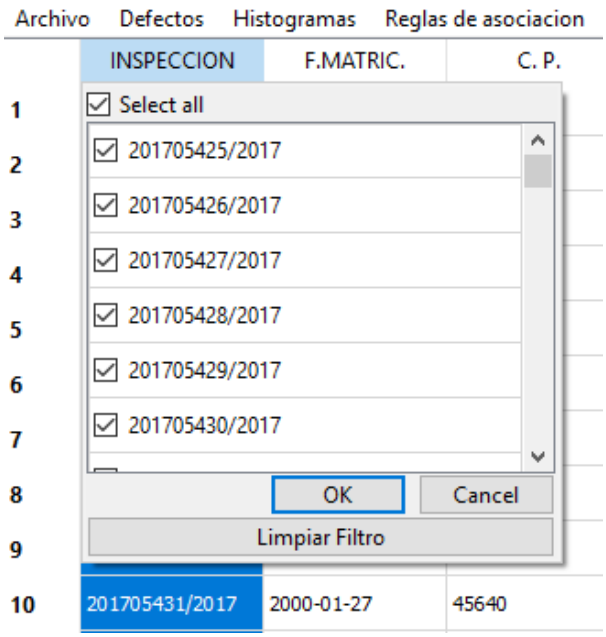


Figura 6.2: Filtrar datos.

6.1.3. Exportar ficheros

Con el fin de poder trabajar con ficheros csv se ha añadido la opción de exportar ficheros para poder transformar los archivos introducidos como .txt en csv y trabajar de una forma más sencilla y fácil con ellos.

6.1.4. Histogramas

La aplicación también tiene la capacidad de crear histogramas en función de marca y modelo, grupo de coche, categoría de coche e inspector.

Una vez seleccionada la opción aparecerá una ventana en la que se puede seleccionar por el tipo del elemento seleccionado, esto generará un histograma en relación entre el número de coches y defectos.

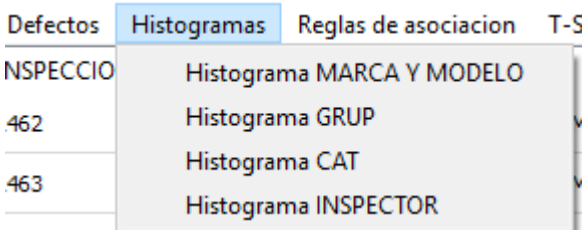


Figura 6.3: Tipos de histogramas.

Un ejemplo de los histogramas que crea la aplicación es el que se muestra en la figura 6.3 el cual agrupa los vehículos por defectos.

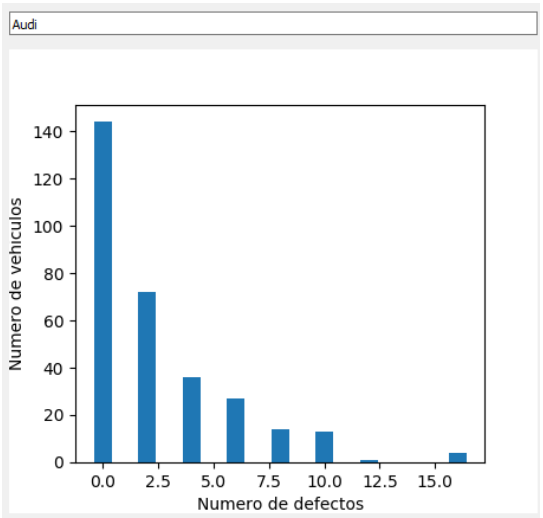


Figura 6.4: Histograma.

Los histogramas son útiles para saber si los datos se agrupan como una distribución normal y ver representado el volumen de defectos. Aunque es cierto que para poblaciones grandes que los datos se agrupen como una distribución normal no es tan relevante, es interesante ver si los datos se agrupan como dicha distribución [45].

Un ejemplo de histograma es el que se muestra en la figura 6.4 el cual hace alusión a los defectos detectados en vehículos de la marca Audi.

6.1.5. Ver defectos

Esta funcionalidad permite ver los defectos detectados por cada inspector junto con la media de defectos detectados. También muestra cuántas veces se detecta un defecto y la proporción que supone respecto al resto de defectos detectados, como se puede apreciar en la figura 6.5.

Esta funcionalidad puede ayudar a ver la proporción de un defecto determinado en los defectos que detecta un inspector.

Inspector | Media	Defecto	Num. Defecto	Proporción
▼ 436.0 | 37.8709677419...			
	401.0-leve	99	0.042163543441226574
	202.0-grave	8	0.0034071550255536627
	407.0-leve	67	0.028534923339011926
	202.0-leve	223	0.09497444633730835
	601.0-leve	154	0.065587734241908

Figura 6.5: Ver defecto.

6.1.6. Reglas de asociación

Las reglas de asociación son una de las partes más cruciales del proyecto ya que nos permiten ver si hay defectos que siempre están relacionados. Estas se pueden calcular en función de marca y modelo, grupo de coche, categoría de coche e inspector.

Una vez seleccionada la opción, aparecerá una ventana en la que se puede seleccionar por el tipo del elemento seleccionado, ya con el elemento seleccionado se obtendrá el cálculo de las reglas, como se muestra en la figura 6.6.

ACM VQ 24 85 SA3

citroen

antecedants	consequents	support	confidence	lift	Nº de elementos
['202.0']	['103.0']	0.258026159334...	0.359447004608...	1.489137590520...	78.0
['103.0']	['202.0']	0.241379310344...	0.384236453201...	1.489137590520...	78.0
['103.0']	['401.0']	0.241379310344...	0.216748768472...	1.274725274725...	44.0
['401.0']	['103.0']	0.170035671819...	0.307692307692...	1.274725274725...	44.0
['103.0']	['901.0']	0.241379310344...	0.463054187192...	1.306807286673...	94.0
['901.0']	['103.0']	0.354340071343...	0.315436241610...	1.306807286673...	94.0
['103.0']	['904.0']	0.241379310344...	0.305418719211...	1.484723369116...	62.0
['904.0']	['103.0']	0.205707491082...	0.358381502890...	1.484723369116...	62.0
['202.0']	['401.0']	0.258026159334...	0.258064516129...	1.517708098353...	56.0
['401.0']	['202.0']	0.170035671819...	0.391608391608...	1.517708098353...	56.0

Figura 6.6: Reglas de asociación.

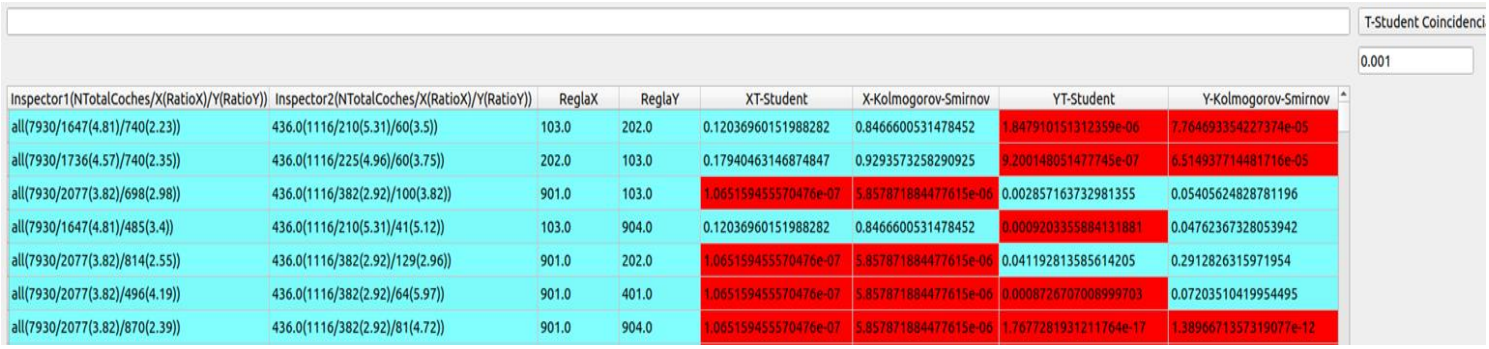
6.1.7. Test estadísticos

Mediante esta funcionalidad podemos realizar tanto el test de Kolmogorov-Smirnov y el test de la t-Student.

En la figura 6.7 se muestra un ejemplo del cálculo de los test estadísticos en la aplicación, esta permite comparar los resultados de inspector en una regla de defectos con todo el conjunto de inspectores o con otro inspector, permitiéndonos saber si existe una anomalía de soporte o confianza si se compara con el conjunto de inspectores, o conflicto de soporte y confianza si se compara con otro inspector.

Es importante destacar que solo se muestran las reglas con p-value más pequeño que el suministrado por el usuario, en este caso 0,001 estos p-value la aplicación los marcara como rojo.

También se intercambia los colores cada vez que cambia el inspector1 entre blanco y azul.



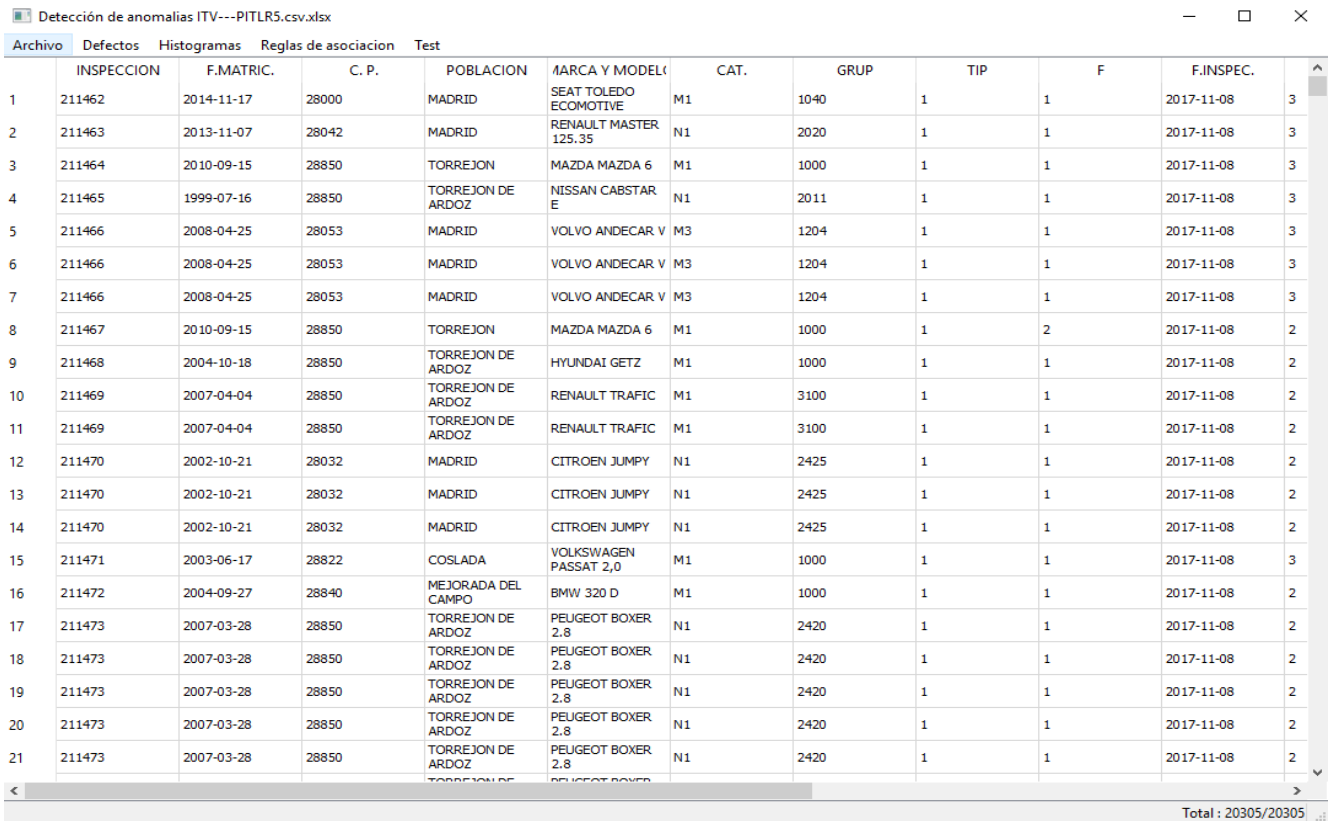
Inspector1(NTotalCoches/X(RatioX)/Y(RatioY))	Inspector2(NTotalCoches/X(RatioX)/Y(RatioY))	ReglaX	ReglaY	XT-Student	X-Kolmogorov-Smirnov	YT-Student	Y-Kolmogorov-Smirnov
all(7930/1647(4.81)/740(2.23))	436.0(1116/210(5.31)/60(3.5))	103.0	202.0	0.12036960151988282	0.8466600531478452	1.8479101513123 59e-06	7.764693354227374e-05
all(7930/1736(4.57)/740(2.35))	436.0(1116/225(4.96)/60(3.75))	202.0	103.0	0.17940463146874847	0.9293573258290925	9.200148051477745e-07	6.514937714481716e-05
all(7930/2077(3.82)/698(2.98))	436.0(1116/382(2.92)/100(3.82))	901.0	103.0	1.065159455570476e-07	5.857871884477615e-06	0.002857163732981355	0.05405624828781196
all(7930/1647(4.81)/485(3.4))	436.0(1116/210(5.31)/41(5.12))	103.0	904.0	0.12036960151988282	0.8466600531478452	0.0009203355884131881	0.04762367328053942
all(7930/2077(3.82)/814(2.55))	436.0(1116/382(2.92)/129(2.96))	901.0	202.0	1.065159455570476e-07	5.857871884477615e-06	0.041192813585614205	0.2912826315971954
all(7930/2077(3.82)/496(4.19))	436.0(1116/382(2.92)/64(5.97))	901.0	401.0	1.065159455570476e-07	5.857871884477615e-06	0.0008726707008999703	0.07203510419954495
all(7930/2077(3.82)/870(2.39))	436.0(1116/382(2.92)/81(4.72))	901.0	904.0	1.065159455570476e-07	5.857871884477615e-06	1.7677281931211764e-17	1.3896671357319077e-12

Figura 6.7: Test estadísticos.

6.1.8. Vistas principales

La vista principal que se muestra una vez entrado en la aplicación y con los datos ya cargados es la que se muestra en la figura 6.8. Si no se carga el fichero aparecerá una imagen de fondo y se podrá acceder a las funcionalidades de la aplicación, pero no se mostrará ningún dato.

Esta ventana permite acceder a todas las funcionalidades descritas previamente y además permite ver el número de filas de elementos que tenemos de datos.



	INSPECCION	F.MATRIC.	C. P.	POBLACION	MARCA Y MODEL(	CAT.	GRUP	TIP	F	F.INSPEC.	
1	211462	2014-11-17	28000	MADRID	SEAT TOLEDO ECOMOTIVE	M1	1040	1	1	2017-11-08	3
2	211463	2013-11-07	28042	MADRID	RENAULT MASTER 125.35	N1	2020	1	1	2017-11-08	3
3	211464	2010-09-15	28850	TORREJON	MAZDA MAZDA 6	M1	1000	1	1	2017-11-08	3
4	211465	1999-07-16	28850	TORREJON DE ARDOZ	NISSAN CABSTAR E	N1	2011	1	1	2017-11-08	3
5	211466	2008-04-25	28053	MADRID	VOLVO ANDECAR V	M3	1204	1	1	2017-11-08	3
6	211466	2008-04-25	28053	MADRID	VOLVO ANDECAR V	M3	1204	1	1	2017-11-08	3
7	211466	2008-04-25	28053	MADRID	VOLVO ANDECAR V	M3	1204	1	1	2017-11-08	3
8	211467	2010-09-15	28850	TORREJON	MAZDA MAZDA 6	M1	1000	1	2	2017-11-08	2
9	211468	2004-10-18	28850	TORREJON DE ARDOZ	HYUNDAI GETZ	M1	1000	1	1	2017-11-08	2
10	211469	2007-04-04	28850	TORREJON DE ARDOZ	RENAULT TRAFIC	M1	3100	1	1	2017-11-08	2
11	211469	2007-04-04	28850	TORREJON DE ARDOZ	RENAULT TRAFIC	M1	3100	1	1	2017-11-08	2
12	211470	2002-10-21	28032	MADRID	CITROEN JUMPY	N1	2425	1	1	2017-11-08	2
13	211470	2002-10-21	28032	MADRID	CITROEN JUMPY	N1	2425	1	1	2017-11-08	2
14	211470	2002-10-21	28032	MADRID	CITROEN JUMPY	N1	2425	1	1	2017-11-08	2
15	211471	2003-06-17	28822	COSLADA	VOLKSWAGEN PASSAT 2,0	M1	1000	1	1	2017-11-08	3
16	211472	2004-09-27	28840	MEJORADA DEL CAMPO	BMW 320 D	M1	1000	1	1	2017-11-08	2
17	211473	2007-03-28	28850	TORREJON DE ARDOZ	PEUGEOT BOXER 2.8	N1	2420	1	1	2017-11-08	2
18	211473	2007-03-28	28850	TORREJON DE ARDOZ	PEUGEOT BOXER 2.8	N1	2420	1	1	2017-11-08	2
19	211473	2007-03-28	28850	TORREJON DE ARDOZ	PEUGEOT BOXER 2.8	N1	2420	1	1	2017-11-08	2
20	211473	2007-03-28	28850	TORREJON DE ARDOZ	PEUGEOT BOXER 2.8	N1	2420	1	1	2017-11-08	2
21	211473	2007-03-28	28850	TORREJON DE ARDOZ	PEUGEOT BOXER 2.8	N1	2420	1	1	2017-11-08	2

Figura 6.8: Ventana principal.

7. Conclusiones y trabajo futuro

En este capítulo recopilamos los resultados obtenidos y proponemos futuras líneas de desarrollo.

7.1. Conclusiones

Este proyecto me ha ayudado a darme cuenta de lo útil e importante que es la minería de datos y de los múltiples campos que tiene para el análisis de datos.

Si bien es cierto que en función del problema hay varias alternativas en cada problema de datos se deben examinar las diferentes posibilidades, para ver cuál se adapta mejor al problema. Conocer estas alternativas, sus posibilidades y limitaciones, resulta fundamental para hallar la mejor solución al problema planteado.

Otra conclusión a la que he llegado es que la fase de acondicionamiento de los datos para aplicar técnicas de minería de datos es muy importante, ya que en función de lo bien que se haga esta fase luego será más fácil aplicar las técnicas correspondientes, mientras que si por el contrario se hace incorrectamente no se obtendrán resultados de calidad. Esto implicara que haya que volver a esta fase repetidas veces para arreglar diversos errores en la preparación de los datos.

Los lenguajes de programación R y Python facilitan mucho la tarea a la hora de hacer cálculos en relación a la estadística, cálculos complejos que a mano pueden llevar bastante tiempo estos lenguajes lo pueden hacer en minutos.

Finalmente concluir que realizar este proyecto ha sido una experiencia muy enriquecedora y gratificante, tanto a nivel académico como personal. He aprendido mucho durante los meses que he trabajado en él, conociendo nuevas tecnologías, desarrollando con lenguajes de programación diferente los aprendidos en nuestra carrera y superando los retos que iban apareciendo en el desarrollo de la aplicación.

Los objetivos que se han cumplido se enumeraran a continuación.

1. Procesamiento de los datos de los ficheros.
2. Entendimiento y uso de las reglas de asociación verificando que se cumplen en el conjunto de datos.
3. Entendimiento e implementación de los test estadísticos en el proyecto.
4. Implementación de una aplicación que resuelva el problema descrito previamente.
5. Ampliación de conocimientos en el campo de la minería de datos.

7.2. Trabajo futuro

En este apartado hablaremos de las posibles mejoras para mejorar la eficiencia de la aplicación para reducir los tiempos de procesamiento.

Mejora en el algoritmo de interacción entre las reglas de asociación y test estadísticos: Las operaciones para encontrar reglas de asociación y realizar los test estadísticos son operaciones bastante costosas hablando en tiempo de ejecución, si se mejora el algoritmo se podrían reducir tiempos de cálculo y usar ficheros o bases de datos más grandes.

Asociación con inspectores: Poder saber si algún inspector tiene especial facilidad para encontrar un defecto determinado, esto permitirá poder investigar qué hace ese inspector y si es posible predecir o automatizar la detección de ese defecto mediante los datos de los que disponemos en la aplicación.

Reglas más complejas en test: Existen reglas de asociación que están formadas por un conjunto de defectos como se muestra en la figura 7.1, estas reglas de asociación son reglas que aparecen en vehículos con múltiples defectos y que relacionan varios defectos en la parte izquierda o derecha con la otra parte de la regla.

En la aplicación estas reglas se obtienen, pero a la hora de usarlas en los test de la t-Student y el test de Kolmogorov-Smirnov no se han implementado por falta de tiempo.

['901.0', '202.0']	['904.0']	0.083937823834...	0.666666666666...	6.245954692556...	54.0
['904.0', '202.0']	['901.0']	0.066321243523...	0.843750000000...	5.287134740259...	54.0
['901.0']	['904.0', '202.0']	0.159585492227...	0.350649350649...	5.287134740259...	54.0
['904.0']	['901.0', '202.0']	0.106735751295...	0.524271844660...	6.245954692556...	54.0

Figura 7.1: Trabajo futuro.

Asociación con fechas y modelo: Verificar si existe alguna fecha en la que los vehículos de un mismo modelo contienen el mismo defecto. Otro factor que se podría verificar es si hay un modelo determinado que siempre presente los mismos defectos así podremos conocer que un determinado modelo en una fecha salió con defectos.

7.3. Conclusions

This project has helped me realize how useful and important data mining is in relation to data analysis.

While it is true that depending on the problem there are several alternatives in each data problem, you should examine these possibilities to see which one fits better the particular problem. Knowing these alternatives, their possibilities and limitations, it is essential to find the best solution to the problem posed.

Another factor that I have drawn in conclusion is that the phase of data preprocessing is very important, since depending on how well this phase is done it will be easier to apply the corresponding techniques, while on the contrary, if it is done incorrectly, quality results will not be obtained. This will mean that you will have to go back to this phase several times to fix various errors in the preparation of the data.

The programming languages of R and Python make it much easier to do calculations in relation to statistics, complex calculations that can take quite a while to do by hand these languages can do in minutes.

Finally, to conclude that this project has been a very enriching and rewarding experience, both academically and personally. I have learned a lot during the months I have worked on it, getting to know new technologies, developing with different programming languages the ones learned in our career and overcoming the challenges that were appearing in the development of the application.

The objectives that have been met will be listed below.

1. Processing of the data in the files.
2. Understanding and use of association rules verifying that they are met in the dataset.
3. Understanding and implementation of statistical tests in the project.
4. Implementation of an application that solves the problem described above.
5. Extension of knowledge in the field of data mining.

7.4. Future Work

In this section we will talk about the features that have not been implemented and possible improvements to improve the efficiency of the application to reduce processing times.

Improvement in the algorithm of interaction between the rules of association and statistical tests: The operations to find rules of association and perform statistical tests are

quite expensive operations speaking at runtime, if the algorithms are improved, calculation times could be reduced, and larger files or databases could be used.

Association with inspectors: Know if any inspector has special ease to find a defect, this will allow to investigate what does that inspector to find it, if it is we want to implement a functionality to predict or automate the detection of that defect through the data available in the application.

More complex rules in test: There are association rules that are formed by a set of defects as shown in figura 7.2; these association rules are rules that appear on vehicles with multiple defects that relate several defects on the left or right side to the other side of the rule.

In the application these rules are obtained, but when we use them in the t-Test and the Kolmogorov-Smirnov test they have not been implemented due to lack of time.

['901.0', '202.0']	['904.0']	0.083937823834...	0.666666666666...	6.245954692556...	54.0
['904.0', '202.0']	['901.0']	0.066321243523...	0.843750000000...	5.287134740259...	54.0
['901.0']	['904.0', '202.0']	0.159585492227...	0.350649350649...	5.287134740259...	54.0
['904.0']	['901.0', '202.0']	0.106735751295...	0.524271844660...	6.245954692556...	54.0

Figura 7.2: Future work.

Association with dates and model: Check if there is a date on which all vehicles of the same model contain the same defect. Another factor that could be verified is if there is a certain model that always presents the same defects, so we can know that if a certain model on a date came out with defects.

Referencias

1. Py2exe. (2011, March 14). Tutorial. Retrieved from http://www.py2exe.org/index.cgi/Tutorial
2. Aaronzira. (2017, Feb 13). Aaronzira/Apriori. Retrieved from https://github.com/aaronzira/apriori/blob/master/apriori.py
3. Hardes, T. Association Rule Mining. Retrieved from http://thardes.de/big-data-englisch/association-rule-mining/
4. Bramer, M. A. (2017). *Principles of data mining*. Springer.
5. Casillas, C., & G. (1970, January 01). Uso de análisis asociativo en algoritmos de aprendizaje. Retrieved from https://repositorio.uam.es/handle/10486/670712
6. Wikipedia. (2018, July 19). Contraste de hipótesis Retrieved from https://es.wikipedia.org/wiki/Contraste_de_hipótesis
7. Berzal, F. Reglas de asociación. Retrieved from https://es.scribd.com/document/356024509/D2-Association-pdf
8. D, K. (2018, March 15). Mining Frequent itemsets - Apriori Algorithm. Retrieved from http://dwgeek.com/mining-frequent-itemsets-apriori-algorithm.html/
9. Field, A. P., Miles, J., & Field, Z. (2014). *Discovering statistics using R*. Sage.
10. Field, A., & Hole, G. (2011). *How to Design and Report Experiments*. Sage.
11. IBM Knowledge Center. Lift in an association rule. Retrieved from https://www.ibm.com/support/knowledgecenter/es/SSEPGG_9.5.0/com.ibm.im.model.doc/c_lift_in_an_association_rule.html
12. Matplotlib development team. (2018, Aug 11). Documentation. Retrieved from https://matplotlib.org/
13. SciPy community. (2018, May 5). Documentation. Retrieved from https://docs.scipy.org/doc/scipy/reference/tutorial/general.html
14. Kim, J. (2014, January 17). Apriori algorithm. Retrieved from https://www.slideshare.net/kujungmul/apriori-algorithm-30118694
15. Moffitt, C. (2017, July 3). Introduction to Market Basket Analysis in Python. Retrieved from http://pbpython.com/market-basket-analysis.html

16. PyQt5.(2018, August 2). Project description. Retrieved from https://pypi.org/project/PyQt5/
17. Pandas. (2018, July 29). Retrieved from https://es.wikipedia.org/wiki/Pandas
18. P. (1970, January 01). Python 3 para impacientes. Retrieved from https://python-para-impacientes.blogspot.com.es/2016/10/calculo-estadistico.html
19. Pitol, F. (1970, January 01). Fermín Pitol, Blog de Inteligencia Artificial en español. Retrieved from http://ferminpitol.blogspot.com.es/2014/05/reglas-de-asociacion-algoritmo-apriori.html
20. Preparando Sublime Text 3 para programar en Python. (2018, March 24). Retrieved from https://hackpuntes.com/preparando-sublime-text-3-programar-python/
21. Pandas. (2018, August 3). Python Data Analysis Library Retrieved from https://pandas.pydata.org/
22. R. (2018, March 26). Rasbt/mlxtend. Retrieved from https://github.com/rasbt/mlxtend/blob/master/paper.md
23. Wikipedia. (2018, June 27) .Reglas de asociación. Retrieved from https://es.wikipedia.org/wiki/Reglas_de_asociación
24. Amat, J. (2016, Jan 1). Pruebas de los rangos con signo de Wilcoxon Retrieved from https://rpubs.com/Joaquin_AR/218464
25. Steinback, Kumar. (2015, August 31). TDT4300: Data Warehousing and Data Mining. Retrieved from https://www.wikipendium.no/TDT4300_Data_Warehousing_and_Data_Mining
26. Tang, J., Chuang, L., Hsi, E., Lin, Y., Yang, C., & Chang, H. (2013). Identifying the Association Rules between Clinicopathologic Factors and Higher Survival Performance in Operation-Centric Oral Cancer Patients Using the Apriori Algorithm. *BioMed Research International, 2013*, 1-7. doi:10.1155/2013/359634.
27. Bernard, Y. Test de Kolmogorov-Smirnov. Retrieved from https://ljk.imag.fr/membres/Bernard.Ycart/emel/cours/ts/node7.html
28. Wikipedia. (2018, July 27). Test t de Student. Retrieved from https://gl.wikipedia.org/wiki/Test_t_de_Student
29. Bernard, Y. Tests Estadísticos. Retrieved from https://ljk.imag.fr/membres/Bernard.Ycart/emel/cours/ts/ts.html
30. Turcios, S., & Alberto, R. (2015,March). T-Student: Usos y abusos. Retrieved from

http://www.scielo.org.mx/scielo.php?script=sci_arttext&pid=S0188-21982015000100009

31. A beginner's tutorial on the apriori algorithm in data mining with R implementation | HackerEarth Blog. (2017, September 15). Retrieved from https://www.hackerearth.com/blog/machine-learning/beginners-tutorial-apriori-algorithm-data-mining-r-implementation/
32. Sinnexus. Datamining (Minería de datos). Retrived from https://www.sinnexus.com/business_intelligence/datamining.aspx
33. Ministerio de Industria, Energía Y Turismo. Junio 2016.Manual_ITV_V710
34. XLSTAT.(2016, May 6).¿Cuál es la diferencia entre pruebas par. Retrieved from https://help.xlstat.com/customer/es/portal/articles/2062456
35. Moore, David; McCabe, George (2003). Introduction to the Practice of Statistics (4 ed.). New York: W.H. Freeman and Co. página. 375. ISBN 9780716796572.
36. XLSTAT. (2017, October 1). ¿Cuál es la diferencia entre una prueba. Retrieved from https://help.xlstat.com/customer/es/portal/articles/2062454-%C2%BFcu%C3%A1l-es-la-diferencia-entre-una-prueba-de-dos-colas-bilateral-y-de-una-cola-unilateral
37. Rumsey, D. (2016, June). What a p-Value Tells You about Statistical Data – dummies. Retrieved from https://www.dummies.com/education/math/statistics/what-a-p-value-tells-you-about-statistical-data/
38. Editor, M. B. What Are T Values and P Values in Statistics? Retrieved from http://blog.minitab.com/blog/statistics-and-quality-data-analysis/what-are-t-values-and-p-values-in-statistic
39. Pereira, O. C. PRUEBA T STUDENT MUESTRAS DEPENDIENTES. Retrieved from https://prezi.com/yptdwwfytmj0/prueba-t-student-muestras-dependientes/
40. El Turrito Cumbierito, H. (2014). KOLGOMOROV-SMIRNOV. Retrieved from https://es.slideshare.net/ElTurritoCumbieritoH/tarea-io
41. Wikipedia.(2018, July 4).Kolmogorov-Smirnov test Retrieved from https://en.wikipedia.org/wiki/Kolmogorov%E2%80%93Smirnov_test.
42. IBM Knowledge Center. Support in an association rule. Retrieved from

https://www.ibm.com/support/knowledgecenter/en/SSEPGG_9.5.0/com.ibm.im.model.doc/c_support_in_an_association_rule.html

43. IBM Knowledge Center. Confidence in an association rule. Retrieved from https://www.ibm.com/support/knowledgecenter/en/SSEPGG_9.5.0/com.ibm.im.model.doc/c_confidence_in_an_association_rule.html
44. Rakesh Agrawal and Ramakrishnan Srikant. 1994. Fast Algorithms for Mining Association Rules in Large Databases. In Proceedings of the 20th International Conference on Very Large Data Bases (VLDB '94), Jorge B. Bocca, Matthias Jarke, and Carlo Zaniolo (Eds.). Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 487-499
45. Bartlett, J. (2013, September 28).The t-test and robustness to non-normality.http://thestatsgeek.com/2013/09/28/the-t-test-and-robustness-to-non-normality/